A Novel Iterative non-Deterministic Algorithm for the Data Clustering Problem

BY

Ahmed Mohd El-Badawi Abdelsatir

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

MAY 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **AHMED ABDELSATIR** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING.**

Thesis Committee

Sadiq Sait. M

Dr. Sadiq M. Sait (Advisor)

Dr. Aiman Helmi El-Maleh (Member)

Tarek El-Bassu

Dr. Tarek Ahmed Helmy El-Bassuny (Member)

Alshayeb

Dr. Mohammad Alshayeb (Member)

Dr. Umair Farooq Siddiqi (Member)

Dr. Adel Fadhl Noor Ahmed

Department Chairman

Dr. Salam A. Zummo

Dean of Graduate Studies

10/1/2018

Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Dedication

To my parents, whose dedication towards the upbringing of my siblings and me was all a son could wish for. To my brothers and sisters who encouraged, challenged and motivated me to accomplish what I am today. To my loving wife who was always there to support and stand by me through the darkness and good times. To my daughter who opens a window of hope every time she smiles.

ACKNOWLEDGMENTS

Before and all thereafter, my greatest gratitude to all almighty Allah for all the blessing he summoned upon me and yet to come. Thanks to my parents, my wife, brothers, sisters and daughter for the encouragement and trust they invested in me. Special thanks go to my mentor and advisor Dr. Sadiq M. Sait whose patience and sincerity I will be forever in debt. Thanks to my committee members Dr. Aiman Helmi El-Maleh, Dr. Mohammad Alshayeb, Dr. Tarek Ahmed Helmy El-Bassuny and Dr. Umair Farooq Siddiqi for their motivation and valuable suggestions. Thanks to COE department and KFUPM for giving me this opportunity.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Ahmed Mohammed Elbadawi Abdelsatir

TITLE OF STUDY: A Novel Iterative non-Deterministic Algorithm for the Data Clustering Problem

MAJOR FIELD: Computer Engineering

DATE OF DEGREE: MAY 2017

Data clustering is an unsupervised classification method which has widespread application in pattern recognition and machine learning. It partitions unlabeled data into different sets. In clustering, the data items that belong to a same cluster are similar to each other and the data items that belong to different clusters are dissimilar from each other. A popular clustering algorithm is K-means. However, the solution quality of K-means Method is dependent on the initial solution. The quality of clustering is determined in terms of a cluster validity index such as sum of squared error (SSE). This work proposes a Simulated Evolution (SimE) based algorithm to solve the clustering problem.

The main features of the proposed algorithm are as follows: (i) Selection of some data items based on their attributes; (ii) A goodness measure based on mean square quantization error (MSE); and (iii) Mutation by altering the assignment of the selected data items. The performance of the proposed algorithm is compared with eleven recent algorithms, namely: K-means, random search, Genetic Algorithm (GA), Simulated Annealing (SA), Ant colony Optimization (ACO), Honey Bee Mating Optimization (HBMO), Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), Gravitational Search Algorithm with K-means (GSA-KM), Black Hole (BH), and Big Bang-Big Crunch (BB-BC). The real-world problems from UCI repository have been used in the experiments. The results show that the proposed algorithm can achieve solution quality which is better than other algorithms. It can also improve the results of K-means clustering method. It was also compared with GSA-KM algorithm which is another heuristic for improving the results of K-means method. It has achieved solution quality which is better than that of GSA-KM.

ملخص الرساله

الإسم: ***أحمد محمد البدوي عبدالساتر***

عنوان الرساله: ***خوارزميه تكراريه غير متوقعه لحل مشكلة تجميع البيانات***

التخصص: ***هندسة الكمبيوتر***

تاريخ الدرجة: ***شعبان ١٤٣٨ - مايو ٢٠١٧***

تجميع البيانات هو طريقة التصنيف غير الخاضعة للرقابة والتي لديها تطبيقات واسعة النطاق في التعرف على الأنماط والتعلم الآلي. فهي تقوم بتقسيم البيانات غير المصنفة في مجموعات مختلفة في عملية التجميع، بحيث تكون عناصر البيانات التي تنتمي إلى نفس المجموعة متشابهة في السمات مع بعضها البعض، وعناصر البيانات التي تنتمي إلى مجموعات مختلفة تختلف عن بعضها البعض في السمات.

أكثر خوارزميات التجميع شعبيةً هي خوارزمية النقطة المركزية الأقرب. ومع ذلك فإن نوعية الحل في هذه الخوارزمية تعتمد على المتوسط الحسابي وهي أكثر تأثراً بالحل الأولي. ويتم تحديد جودة التجميع عن طريق مؤشر الصلاحية كمجموع الخطأ التربيعي . يقترح هذا العمل خوارزميه مبنيه على خوارزمية محاكاة التلدين المتطورة المستوحاة من علم الفيزياء لحل مشكلة التجميع.

وفيما يلي الخصائص الرئيسية للخوارزمية المقترحة:

'١' اختيار بعض عناصر البيانات استنادا إلى سماتها.

'٢' قياس الجوده إستناداً إلى خطأ متوسط الكميات المربع .

'٣' الطفرة عن طريق تغيير سمات عناصر البيانات المختارة.

تمت مقارنة الخوارزمية المقترحة مع إحدى عشرة خوارزمية تحسين حديثة وهي خوارزمية النقطة المركزية الأقرب ،خوارزمية البحث العشوائي, الخوارزمية الجينية ، خوارزمية محاكاة التلدين، خوارزمية مستعمرة النمل، خوارزمية عسل النحل ، خوارزمية أسراب الطيور، خوارزمية بحث الجاذبية ، خوارزمية بحث الجاذبيه المدمجة مع خوارزمية النقطة المركزية، خوارزمية الثقب الأسود و خوارزمية الارتداد العظيم.

وقد طبقت الخوارزمية بمعايير المقارنة العالمية من جامعة كاليفورنيا. وأظهرت النتائج أن الخوارزمية المقترحة حققت جودة حل أفضل من الخوارزميات الأخرى. كما أمكن أيضا تقليل زمن تنفيذ الخوارزمية بدمجها مع خوارزمية النقطة المركزية الأقرب. كما قمنا أيضا بمقارنة الخوارزمية المدمجة مع خوارزمية بحث الجاذبية المدمجة بخوارزمية النقطه المركزيه الأقرب وتم تحقيق جودة حل أفضل.

CHAPTER 1

INTRODUCTION

1.1 Definition of Clustering

Linguistic experts define clustering as "the number of items of the same kind held together "[1]. From this definition, the clustering depends on identifying similarity among items and based on that similarity objects are grouped. Main function in Clustering process is to identify which items are similar to each other based on a similarity criterion.

Cluster analysis can be described as a statistical classification technique that divides objects into groups (clusters) performed in such a way that objects in one cluster are similar to each other and dissimilar to objects in other clusters.

Clustering is described as an exploration tool that outputs the patterns, association and relationship between dataset objects. Alternatively, Clustering can also be defined as the unsupervised grouping of objects

based on the similarity measures. Clustering process requires a prior knowledge of the number of clusters and based on that centroids are calculated. Then objects are assigned to the closest cluster. The closest cluster for an object is the one which has the nearest distance.

Different distance measures can be used to determine the distance between objects and centroid such as: Euclidean distance, cosine similarity, correlation, etc. [2].

The clustering methods use similarity and dissimilarity based techniques using the attributes of the objects. The objects that have similar attributes are grouped together, whereas the objects that belong to different clusters have dissimilar attributes. The attributes can be related to the functionality of objects. Problem definition can have significant variants, depending on the model specification, for example, a distance-based model will use a traditional distance measure for quantification, whereas a generative model may define similarity on the basis of a probabilistic generative mechanism [3].

Due to an increase in the amount of data and applications that use clustering, the need for efficient and effective clustering methods has been increased. In recent years, the clustering problem has been studied from different aspects.

The application domain is vast and diverse and includes text, multimedia, social networks and biological data analysis, etc. The nature of the clustering in different domains varies greatly from each other.

1.2 Applications of Data Clustering

Clustering explores a diverse type of applications which we will describe briefly in the below mentioned paragraphs.

1. Intermediate Step in data mining problems: Clustering in some aspect is considered as data summarization which is an intermediate step in data mining techniques.
2. Multimedia Data Analysis: Multimedia data includes images, audio and video. The process of the detection of similar segments of videos and photos is performed with the help of clustering.
3. Biological Data Analysis: Recently, the technology of microarray has been evolved and it becomes possible to simultaneously observe the level of thousands of data expressions of genes when they undergo through some specific conditions [4, 5]. There are two main groups of gene expression analysis applications: identification of groups of genes that have closely related expressions and the discovery of new subgroups of pathologies.
4. Social Network Analysis: In these applications, clustering is mainly used for important community detection on the underlying social networks. Social network summarization is also an application of clustering [3].

1.3 Data Clustering Methods

There are many types of clustering methods, but only partitional and hierarchical methods of clustering are widely used. These two methods are widely used because of their ease of implementation and simplicity as compared to the other methods.

1.3.1 Hierarchical Clustering

A hierarchical clustering algorithm performs clustering by repeating cycles of partitioning the large clusters into smaller ones or merging the small clusters into a larger one.

In both ways that process produces a hierarchy of cluster which is called dendrogram. Agglomerative clustering uses the bottom-up approach and the divisive clustering using the top-down approach. For numeric data, some commonly used similarity measures are Euclidean distance, Manhattan distance, and cosine distance. The Hamming distance is usually used in non-numeric data.

The dendrogram is built with the help of these distance metrics. Moreover, in Hierarchical clustering the observations (instances) are not required for building the visual representation of the clusters (dendrogram). In the below mentioned example, we will show the steps in Hierarchical clustering using 6 samples.

TABLE 1.1 depicts the dissimilarity among the 5 objects (samples) where zero indicates the maximum similarity and one indicates the minimum similarity.

Object	A	B	C	D	E	F
A	0	**0.2**	0.3	0.4	0.5	1
B	**0.2**	0	0.6	0.7	0.8	1
C	0.3	0.6	0	0.5	0.6	0.8
D	0.4	0.7	0.5	0	0.8	1
E	0.5	0.8	0.6	0.8	0	1
F	1	1	0.8	1	1	0

TABLE 1.1: Dissimilarity Values for the Objects.

On the first step, the algorithm looks for the most similar pair's step by step. A and B are the first closest pair based on the lowest dissimilarity value 0.2, hence A and B are merged on the level of 0.2 on the dendrogram.

TABLE 1.2 depicts the first step in dendrogram and dissimilarity value recalculation based on the maximum method.

Object	(A,B)	C	D	E	F
(A,B)	0	0.6	0.7	0.8	1
C	0.6	0	**0.5**	0.6	0.8
D	0.7	**0.5**	0	0.8	1
E	0.8	0.6	0.8	0	1
F	1	0.8	1	1	0

TABLE 1.2: First Step on Dendrogram.

On the second step, the same procedure on step one is repeated with

next similar samples. C and D are the next similar samples with 0.5 values.

TABLE 1.3 depicts the second step in dendrogram and dissimilarity value recalculation based on C and D merge.

Object	(A,B)	(C,D)	E	F
(A,B)	0	**0.7**	0.8	1
(C,D)	**0.7**	0	0.8	1
E	0.8	0.8	0	1
F	1	1	1	0

TABLE 1.3: Second Step on Dendrogram.

At this point 4 clusters process performed with dissimilarity value of 0.5 , these clusters are (A, B), (C, D), E, and F.

The process can be repeated based on the dissimilarity degree required. TABLE 1.4 and 1.5 Explain the remaining steps.

Object	(A,B,C,D)	E	F
(A,B,C,D)	0	**0.8**	1
E	**0.8**	0	1
F	1	1	0

TABLE 1.4: Third Step on Dendrogram.

Object	(A,B,C,D,E)	F
(A,B,C,D,E)	0	1
F	1	0

TABLE 1.5: Final Step on Dendrogram.

Dendrogram on the third step where the dissimilarity reaches the value of 0.7 show three clusters (A, B, C, D), E, and F as appeared in TABLE 1.4.

One step after the dendrogram end up with 2 clusters (A, B, C, D, E) and F when dissimilarity value equal to 0.8 as shown on TABLE1.5.

1.3.2 Partitional Clustering

Partitional clustering is about generating different partitions and evaluating them using a cluster validity index. It is also referred to as non-hierarchical method where each object is placed in only one cluster.

The partitional methods require the user to define the number of clusters called (*K*). One of the most common partitional methods is the K-means clustering algorithm.

Partitional clustering is classified as non-over lapping clustering. Partitional clustering most probably works on document data and data which have continuous nature, where a hierarchal method works on categorical data. Section 1.4 describes the partitional clustering in detail.

1.3.3 Hierarchical Vs. Partitional

The main strength of partitional clustering algorithms is that they can improve solution quality through iterative optimization. This is not the case in standard hierarchical clustering where dendrogram do not have the ability to revisit the merges (or splits) that were already completed. Partitional clustering is also computationally efficient as compared to hierarchical clustering.

The visual dendrogram provides a live assistance during clustering process and is considered to be one of its greatest advantages [3]. Compared to the non-deterministic nature of K-means, the Hierarchical methods are classified as a deterministic method.

Dendrogram inability to revisits the merges or splits is mitigated on later versions of hierarchical clustering by sacrificing the run time complexity, years after the complexity also reduced to linear time by taking the advantages of Parallel hierarchical clustering methods. FIGURE 1.1 and TABLE 1.6 describe and compare between standard hierarchical and partitional method.

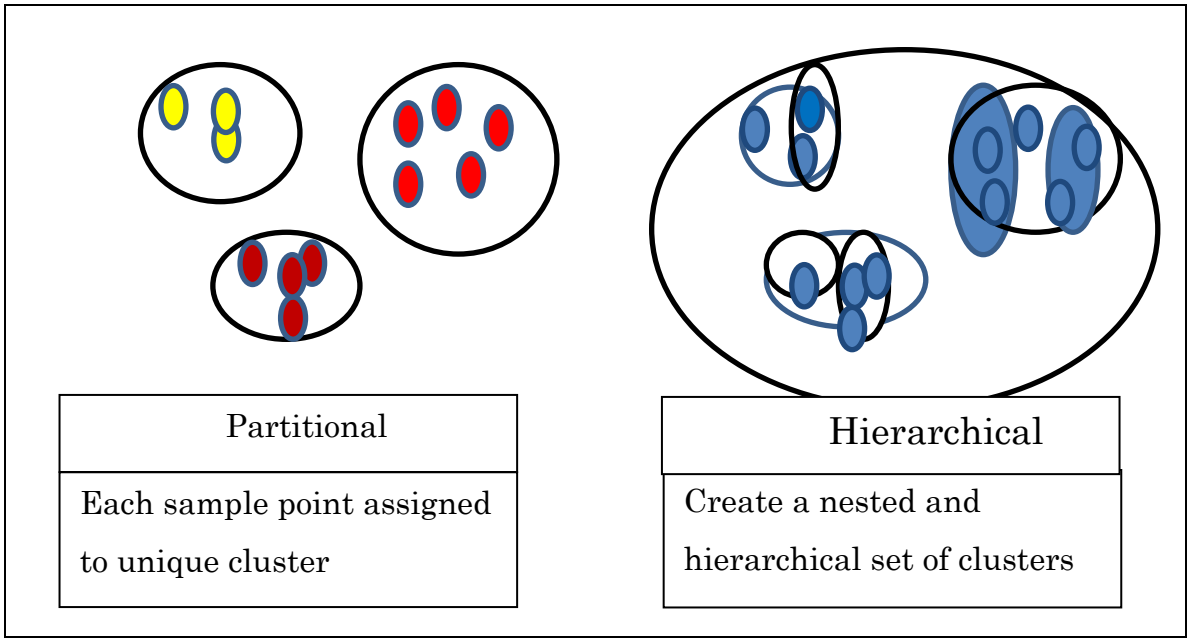

FIGURE 1.1: Partitional Vs. hierarchical.

Method	*Run Time*	*Nature*	*clusters*	*Flexible*	*Pre-defined No of clusters*
Hierarchical	Complex	Deterministic	Overlapping	No	No
Partitional	Non-complex	Non-deterministic	Non-overlapping	Yes	Yes

TABLE 1.6: Comparison between Hierarchical and Partitional Method.

1.4 K-means

K-means algorithm is a partitional method of clustering and requires that the number of clusters (*K*) should be known a priori. It is an iterative algorithm. In the first iteration, the centroids (center of clusters) are randomly initialized. In each iteration, the following two operations are performed: (i) Assign each object to the cluster whose centroid is closest to it; and (ii) Re-calculate the centroid of each cluster based on the objects currently assigned to it.

The K-means method terminates when the centroids of clusters stop changing their values. In K-means method, the solution quality is highly dependent on the initial value of centroids. The most important disadvantages of K-means algorithm are: (i) Its inability to find the global optimum solution in most cases, and (ii) Dependency on the initial values. It is also non-deterministic and yields different results in different runs. K-means Steps are explained in FIGURE 1.2.

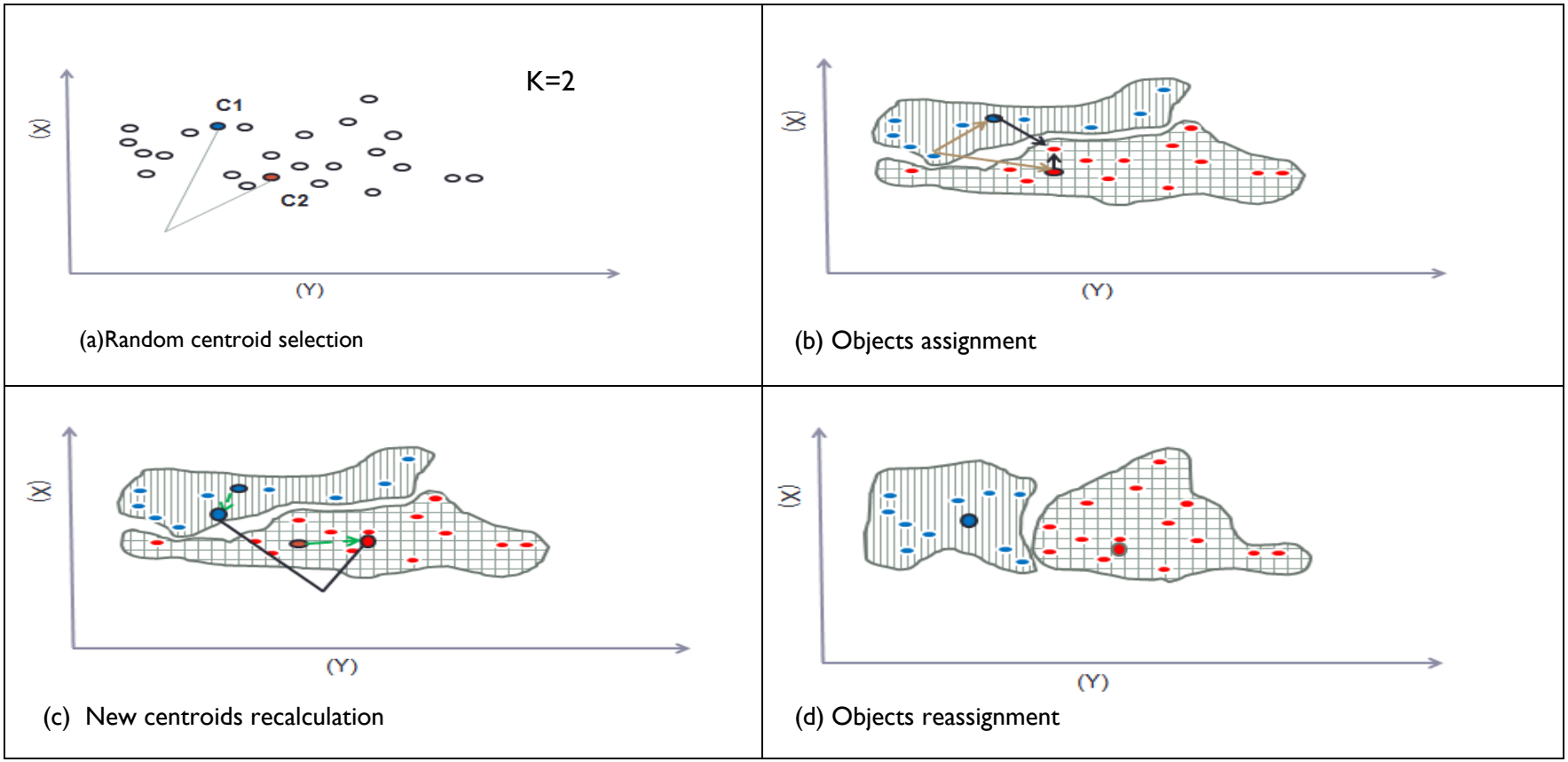

FIGURE 1.2 : Standard K-means Steps for two Clusters.

1.4.1 K-means Complexity Order

K-means convergence happened on the first iterations. The Algorithm complexity is $O(n * k * i * d)$

$$Where \quad n = number\ of\ objects(data\ points)$$

$$k = number\ of\ clusters$$

$$i = number\ of\ iterations$$

$$d = number\ of\ attributes$$

1.4.2 Limitation of K-means

K-means has some limitation which urges the study of implementing other algorithms to overcome below K-means limitations.

1. Use clusters of unequal sizes, whereas, the K-means method performs well when the clusters have equal sizes.
2. Different density of clusters, in this case, clusters can be equal in size while object distribution is totally variant.
3. Non spherical shape of cluster.
4. Outliers or object that appears to be noise.

1.4.3 Mathematical Model of Data-Clustering

Clustering is to partition a dataset of $n\ objects\ \boldsymbol{O} = \{O1, O2, O3, \ldots . On)$ into *K* clusters and each object consists of $\boldsymbol{l}\ attributes$.

The clusters are represented as (C1, C2, C3… ,Ck). The objective of clustering is to group the similar and relative objects together. The clustering should meet the below mentioned conditions:

(a) No empty cluster (each cluster must have at least one object) i.e.

$$\mathrm{Ci} \neq \emptyset \quad \forall \mathrm{i} \in \{1,2,3, \ldots, \mathrm{k}\}$$

(b) One object can't be a member in two clusters i.e. $\boldsymbol{Ci} \cap \boldsymbol{Cl} = \boldsymbol{0}$

(c) Each object should be assigned to a cluster. In other words the sum of objects in all clusters should equal the size of the dataset.

Different methods of clustering can be developed if they meet the mentioned three conditions. The quality of clustering is determined using a cluster validity index. In this work, we use the most widely used index through the literature Mean Square Error (MSE).

1.5 MSE Measure

1.5.1 MSE Definition

Mean square quantization error or (MSE) is a measure of the average of the squares of the errors or deviations. The main application for MSE is the process of analog to digital conversion. In this process the analog signals in continuous range of values are converted to discrete set of values by comparing them with a sequence of thresholds [6].

In our problem the analog value is considered to be the attributes of the objects and the thresholds value are considered to be the clusters centroids, the smaller the value of the MSE the better the solution is. MSE measure has been used in different applications such as speech analysis and recognition, analog to digital conversion, noise identification/reduction process and data clustering [7].

1.5.2 MSE Representation for Clustering

$$\mathrm{MSE} = \sum_{i=1}^{K} \sum_{\forall\, Oi \in Ci} d(Oi, Ci)^2 \qquad (1.1)$$

Where the $d(Oi, Ci)$ is the distance between the object Oi and cluster centroid Ci which the object is assigned to, K is the number of clusters, the objective here is to minimize the distance measured by Euclidian distance, the formula below defines the Euclidian distance for two different objects Oi, Oj.

$$d(Oi, Oj) = \sqrt{\left(\sum_{p=1}^{l} \left(O_i^p - O_j^p\right)^2\right)} \quad (1.2)$$

$\forall\, l = number\ of\ dimensions$

Where $\boldsymbol{p}$ is the attribute and $\boldsymbol{l}$ is the number of dimension.

1.6 Thesis Contribution

This work used iterative non-deterministic Algorithm for clustering with the main objective to find the optimal solution (cluster centroids) and assignments of objects to those centroids with reasonable runtime, assuming the number of clusters is given. Below are the tasks achieved in this study.

1. Selected suitable iterative non-deterministic algorithm (SimE).
2. Formulated the optimization problem, identified the objective and cost function of the algorithm.

3. Customized selection and allocation operations of the algorithm. Operator and parameter tuned to the best value to obtain best solutions
4. Evaluated the performance of proposed algorithm on five standard benchmarks.
5. Compared the proposed algorithm with recent methods solving the same problem.

1.7 Thesis Organization

The remaining of this thesis is organized as follows: Chapter 2 discusses the formal description and similarity measures; Chapter 3 explores the related works from literature. Chapter 4 details the SimE stages and describes our method implementation and design; Chapter 5 shows the experimental results and configuration. In Chapter 6 we conclude with future works and summary of the results.

CHAPTER 2

FORMAL DESCRIPTION OF CLUSTERING

The problem of data clustering can be formally described as follows: Consider a data-set that contains $\boldsymbol{N}$ objects. Each object consists of $\boldsymbol{D}$ dimensions (or attributes). The attributes represent an observation of the object in one condition. The goal of the proposed algorithm is to find centroids of optimal clusters, when each object in the data-set is assigned to the cluster whose centroid is nearest from it. The number of clusters is represented by K and should be known a priori. The clustering is performed with the objective of minimizing the Total Within-Cluster Variation (TWCV) or square-error [5]. Clustering often aims to maximize the intra-cluster similarity and minimize the inter-cluster similarity among the patterns. In other words, maximization of single cluster homogeneity and different clusters heterogeneity. Most of the clustering methods employ Euclidean distance measure to determine similarity and dissimilarity.

2.1 Computational Complexity

The problem of finding an optimal solution by partitioning $\boldsymbol{N}$ objects into $\boldsymbol{k}$ clusters is NP-complete [8]. Provided that the number of distinct partitions of N objects into k clusters increases approximately as $\frac{\boldsymbol{K^N}}{\boldsymbol{k!}}$, attempting to find a globally optimal solution is usually not computationally feasible [9].

2.2 Similarity Measures

Similarity or proximity is the measure which identifies how relatively the data are closed to each other.

These measures are selected based on the application where clustering is applied. Similarity measures are technically defined as quantification of the similarity between two objects in a real-value [10]. Below we will describe in detail widely used measures in literature.

Block Distance is known as Manhattan measure, city block distance, L1 distance and absolute value distance. For two objects it is defined as the sum of the differences of their corresponding components [11].

The mathematical formula of city distance for $\boldsymbol{a} = \{\boldsymbol{a1}, \boldsymbol{a2}, \boldsymbol{a3}, \ldots, \boldsymbol{an}\}$ $\boldsymbol{and}$ $\boldsymbol{b} = \{\boldsymbol{b1}, \boldsymbol{b2}, \boldsymbol{b3}, \ldots, \boldsymbol{bn}\}$ where $\boldsymbol{n}$ is the number of features. City distance is represented in Equation 2.1.

$$\boldsymbol{d1}(\boldsymbol{a}, \boldsymbol{b}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{ai} - \boldsymbol{bi}) \tag{2.1}$$

Cosine similarity is a similarity between two vectors of the inner space product measuring the cosine of the angle between them. The mathematical formula of Cosine similarity for $\boldsymbol{a} = \{\boldsymbol{a1}, \boldsymbol{a2}, \boldsymbol{a3}, \ldots, \boldsymbol{an}\}$ $\boldsymbol{and}$ $\boldsymbol{b} = \{\boldsymbol{b1}, \boldsymbol{b2}, \boldsymbol{b3}, \ldots, \boldsymbol{bn}\}$ where $\boldsymbol{n}$ is the number of features. Cosine similarity is represented in Equation 2.2.

$$\boldsymbol{cos}(\emptyset) = \frac{(\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{ai} \,.\boldsymbol{bi})}{\sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{ai^2}} \cdot \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{bi^2}}} \tag{2.2}$$

Euclidean distance or L2 distance when the square difference between two vector elements summed and squared that simply represents the Euclidean distance. This measure is widely used in a variety of applications. The mathematical formula of Euclidean distance for $\boldsymbol{a} = \{\boldsymbol{a1}, \boldsymbol{a2}, \boldsymbol{a3}, \ldots, \boldsymbol{an}\}$ $\boldsymbol{and}$ $\boldsymbol{b} = \{\boldsymbol{b1}, \boldsymbol{b2}, \boldsymbol{b3}, \ldots, \boldsymbol{bn}\}$ Where $\boldsymbol{n}$ is the number of features. Euclidean distance is represented in Equation 2.3.

$$d(a, b) = \sqrt{\sum_{i=1}^{n}(ai - bi)^2} \quad (2.3)$$

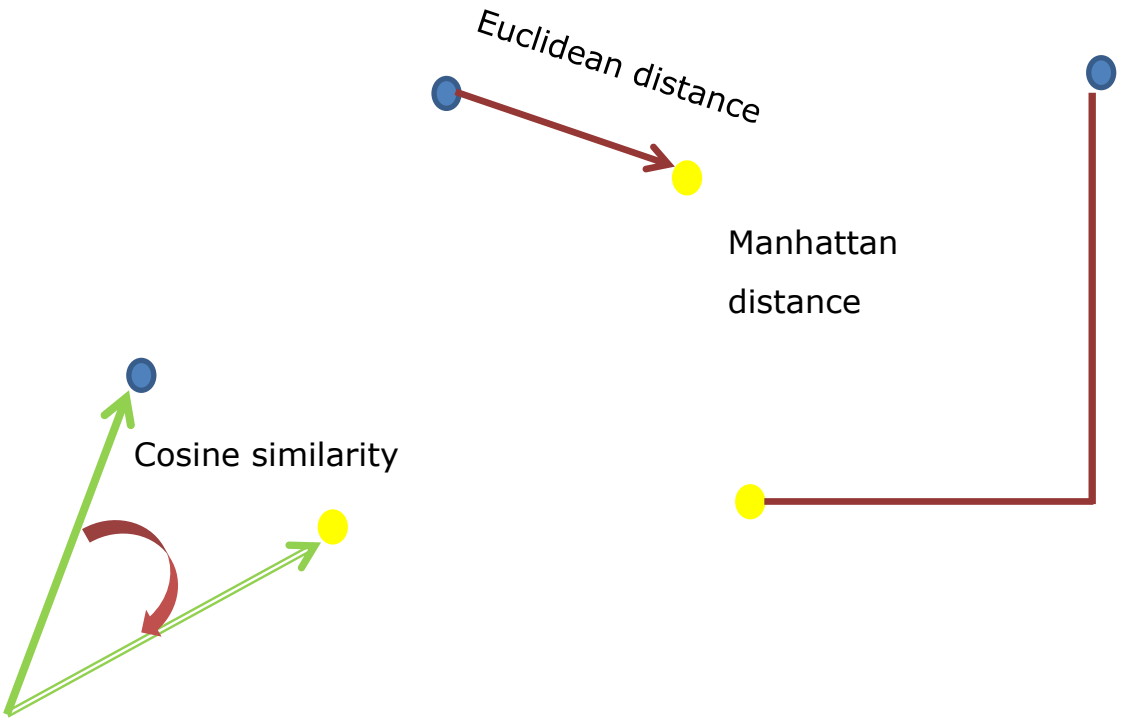

FIGURE 2.1: Similarity Distance Measures.

FIGURE 2.1 explains the distance measure estimation for Euclidean, Manhattan and cosine similarity.

Pearson proximity measure: It identifies the linear correlation between sequences. Any two objects can be represented as real-valued $a = \{a1, a2, a3, \dots, an\}\ and\ b = \{b1, b2, b3, \dots, bn\}$ where $\boldsymbol{n}$ is the number of features. The distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ can be determined as follows:

$$\boldsymbol{distance}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{1} - \boldsymbol{PE}(\boldsymbol{a}, \boldsymbol{b}) \tag{2.4}$$

The Pearson is described on the following equation.

$$\boldsymbol{PE}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{ai} - \boldsymbol{a`})(\boldsymbol{b} - \boldsymbol{b`})}{\sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{ai} - \boldsymbol{a`})^{\boldsymbol{2}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{bi} - \boldsymbol{b`})^{\boldsymbol{2}}}} \tag{2.5}$$

Where $\boldsymbol{a`}$ is the mean of object $\boldsymbol{a}$ and $\boldsymbol{b`}$ is the mean of object $\boldsymbol{b}$.

Pearson measure is used mainly for gene clustering where two genes exhibit to the similarity in shapes or trend, rather than absolute difference from their values, therefore correlation coefficient is employed to determine the similarity between genes. The Pearson correlation coefficient is considered as de-facto proximity measure in gene expression data analysis.

2.3 Cluster Validity indices (CVI)

As we discussed earlier, there are a variety of clustering methods which lead to different clustering solutions. The Cluster Validity Index (CVI) is introduced to evaluate and find the best partition which fits to the data and determines solution quality.

The quality of clustering of two-dimensional data can be easily evaluated by a reader through visual representation. However, for high dimensional data visual validity is not a trivial work and mathematical methods become necessary.

Solution evaluation process for clustering algorithm is called the cluster validity assessment; mainly two criteria's are used together or individually for building the objective function of clustering [11].

- Compactness: members of each cluster should be close to each other. We call it mathematically intra-cluster distance, and by this criteria the clustering technique has to maintain and increase the cluster homogeneity
- Separation: members of different cluster should be widely separated. We call it mathematically inter-cluster distance, and by this criteria, the clustering technique has to maintain the heterogeneity between different clusters members.

CVI also have three categories of validation.

- Internal validity
- External validity
- Relative validity

Internal and external are statistical modules of the validity; their major drawback is their high computational cost [12].

Internal is a validity based on metric of clustering scheme and dataset itself and do not require a prior information about the data set itself.

External validity based on the user specific intuition and previous knowledge of dataset.

Relative validity is a comparison of different clustering scheme, one or more clustering algorithms with different runs and configurations on the same dataset; relative validity aims to find the best clustering way from different results.

On sections 2.3.1 and 2.3.2 we will have a detailed description and examples of some high-rank validity indices of both types internal and external.

2.3.1 Internal Validity Indices

As mentioned above, the internal validity indices work without a prior knowledge of the data, however, they are still the most accurate

Index [13].

Some of the internal indices are discussed below, such as Silhouette index, SD index and S-Dbw.

Silhouette index measures the compactness and the separation between groups. Silhouette index is represented in below equation.

$$\boldsymbol{S(i)} = \frac{\boldsymbol{(b(i)-a(i))}}{\boldsymbol{Max\{a(i),b(i)\}}} \quad (2.6)$$

Where $\boldsymbol{a(i)}$ is the average distance of $(\boldsymbol{i})$ with all other data within the same cluster; $\boldsymbol{b(i)}$ is the lowest average distance of $(\boldsymbol{i})$ to all points in any other cluster.

SD index is defined as the total separation among clusters and the average scattering for clustering. SD mathematically is represented in Equation 2.7.

$$\boldsymbol{SD(nc)} = \boldsymbol{w} \cdot \boldsymbol{Scat(nc)} + \boldsymbol{Dis(nc)} \quad (2.7)$$

The first term $\boldsymbol{Scat(nc)}$ indicates the average compactness of clusters, the small value of this term indicates compact cluster, and the high value indicates scattering within a cluster is high. Second term $\boldsymbol{Dis(nc)}$ indicates the total separation between clusters $\boldsymbol{nc}$ and influenced by the geometry of cluster and increase the number of clusters [14]. Where $\boldsymbol{w}$ is a weighted factor because both terms are in different range to keep them balanced.

S_Dbw index as the SD index; it measures Intra and inter–cluster variance in addition to the density variation among cluster for achieving the more reliable solution.

$$\boldsymbol{S_{Dbw}(nc) = Scat(nc) + Dens_{bw(nc)}} \qquad (2.8)$$

2.3.2 External Validity Indices

As mentioned above the internal validity index does not require a prior knowledge about the dataset where the external requires such knowledge.

Purity and F-measure external indices will be discussed in this section.

Purity is a measure of the amount of truly classified objects per cluster over the sample size. Purity formula is described in Equations 2.9.

$$\boldsymbol{Pj = \frac{1}{mj}\left(Max(i)\left(m_j^i\right)\right)} \quad (2.9)$$

Where $\boldsymbol{mj}$ is the total count of objects in $\boldsymbol{Cluster\ (j)}$ and $\boldsymbol{m_j^i}$ is the total count of objects which are assigned to $\boldsymbol{Cluster\ (j)}$and should be in $\boldsymbol{Cluster\ (i)}$.

The overall purity index calculation is based on weighted sum of all clusters by Equation 2.10.

$$\boldsymbol{Purity\ = \sum_{j=1}^{K} \frac{mj}{m}\ Pj} \quad (2.10)$$

Where $\boldsymbol{K}$ is the total number of clusters and $\boldsymbol{m}$ is the total number of objects.

F-measure combines the precision and recall concepts from information retrieval. We then calculate the recall and precision of that cluster for each class as in Equations 2.11 and 2.12.

$$\boldsymbol{Recall\ (i,j) = \frac{m\,ij}{mi}} \quad (2.11)$$

$$\boldsymbol{Precision\ (i,j) = \frac{m\,ij}{mj}} \quad (2.12)$$

Where $\boldsymbol{mij}$ is the total count of class $\boldsymbol{i}$ objects which assigned to cluster $\boldsymbol{j}$, $\boldsymbol{mj}$ and $\boldsymbol{mi}$ is the total counts of cluster $\boldsymbol{j}$ and class $\boldsymbol{i}$ objects. The F-measure of cluster and class is given by Equation 2.13.

$$\boldsymbol{F(i,j) = \frac{2Recall(i,j)Precision(i,j)}{Precision(i,j) + Recall(i,j)}} \qquad (2.13)$$

CHAPTER 3

LITERATURE REVIEW

3.1 Related Works

The literature was surveyed based on two criterions.

- The algorithms used to solve the clustering problem.
- Similarity and dissimilarity measures employed by the clustering algorithms.

K-means is the most popular method used in data clustering analysis because of the easy implementation and the high computation performance, in spite of that sensibility to the initial solution is considered a weakness for the algorithm which opens the area of research for other algorithms.

Malik U et al. [15] proposed a genetic algorithm, whose main idea is to simulate nature evolution process, which evolves the solution generation after generation, contradicting with the K-means algorithm that might be stuck on the local optimum. Genetic algorithm is not sensible to the initial solution and always escapes from the local optimum solution.

The main drawback of the genetic algorithm is its expensive computation nature which impedes the algorithm from a wide vital application such as bioinformatics data analysis.

Krishna and Murty in [16] proposed a clustering algorithm called Genetic K-means Algorithm (GKA) that interbreeds a K-means algorithm with a genetic algorithm. This crossbreeding method combines genetic algorithm robustness and K-means high performance. As a result, GKA convergence time to the global optimum is faster than standard genetic algorithm.

Lu Y et al. [17] came up with an extension of GKA with incremental Genetic-Means Algorithm (IGKA), the algorithm inherits GKA advantage. This algorithm outperform GKA at a low value of the mutation probability as indicated by the author $\mathbf{0.005}$ for small data and $.\mathbf{0005}$ for huge data that give a clue the performance gained in IGKA depends on the number of patterns that change their clusters, for huge test data even low Probability of mutation may cause many objects to change their membership. The main idea of this algorithm intends for calculating the TWCV (total within-cluster variation) and clustering centroids inclemently. The result shows that IGKA on microarray data has more tendencies to obtain the best result by clustering the similar category genes on the same cluster.

Hruschka et al. [5] solved the problem of gene clustering by evolutionary algorithms which auto estimate the correct number of partitions using a genetic algorithm in partitioning technique and eliminating the crossover operator. He implemented the K-means algorithm as local search and the similarity measure which is used in his approach is the Euclidean distance.

Julia et al. [18] reformulated the problem as multi-objective clustering. The multi-objective clustering with automatic k-determination (MOCK) outperforms the previous technique for single objective (K-means) where the genes nature is diverse. Single objective techniques are suitable for spherical a well-separated clusters, but for complicated cluster structures this technique may not perform well. MOCK used three measures to judge the solution quality based on the different classifications of cluster (connectedness, compact and spatial separation). Authors claimed that the tradeoff between multi-objective functions came up with better results compared to single objective traditional algorithm and ensemble method across diverse range of benchmark data sets. FIGURE 3.1 displays the classification mentioned.

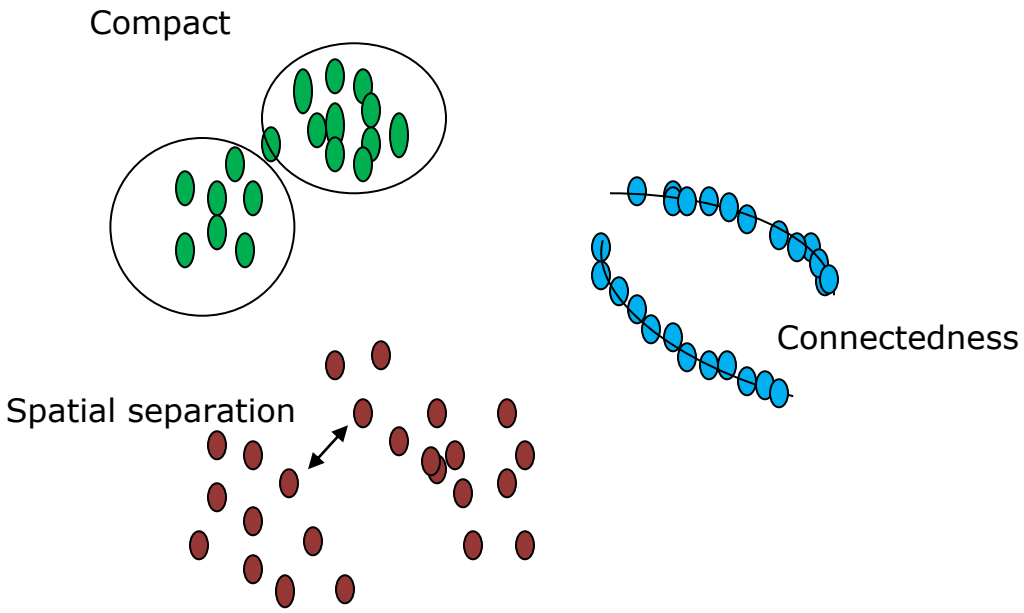

FIGURE 3.1: Three Different Classifications of Clustering Objectives.

Sanghamitra et al. [19] presented a new method significant multi-class membership–two stage (SIMM-TS) working on the data points itself by defining the points that have pertinence to more than one cluster. The first stage is to determine how many clusters and SIMM points based on the result of Variable String length genetic algorithm (VGA). In the second stage re-clustering is done using multiple objective Genetic algorithms (MOGA) after excluding SIMM points, the nearest neighbor criterion is used to assign points. Solutions are evaluated both quantitatively and by visualization tools. The quantitative evaluation is performed by adjusted Rand index and Silhouette index. In order to establish significant statistical improvement of the results produced by the proposed technique, statistical test also comes into the picture. Finally, biological interpretation of the clustering solutions also contributed to the evaluation process. (SIMM-TS) outperforms Self-Organizing Map (SOM) and a recently developed weighted Chinese restaurant-based clustering method (CRC).

Yau-King et al. [20] presented Particle Swarm Optimization based K-means with clustering match (PSO-KM with CM2). This method injected two steps of CM before PSO updating, first by using the nearest distance the algorithm matching cluster sequences that contained in particles with reference to cluster on the global best particle. The second step is a rearrangement process for the first step sequence of clusters. The new method optimizes the cluster sequence for each particle to prepare for next PSO updating process and in return this method to

some extent eliminates the sensitivity to the initial conditions which was one of the problems of traditional K-means approach. (Lymphoma data, yeast cell-cycle data, and sporulation data) are tested in this approach to evaluate algorithm performance, It outperforms traditional K-means, PSO-KM.PK-means and original PSO-KM (CM) in terms of fast convergence and compactness within the cluster.

Shiquan et al. [21] work on the gene expression dataset using particle Swarm optimization (PSO) with a gravitational search algorithm (GSA) (PSOGSA). They integrate the exploration ability in GSA to exploitation in PSO for updating velocity equations. Position equations are updated through a mobility factor, for improving PSOGSA accuracy and convergence diversity of population held the responsibility of guiding the mobility factor. Accuracy and mutual information are the basis of comparison in four methods including PSOGSA, *K-means*, Pure PSO and pure GSA. They experimented three cancer and four artificial datasets. PSOGSA achieved the highest score as compared to other algorithms in all datasets except for artificial ones. As an example, mutual information values for colon cancer by GSA, K-means, and PSO are $\mathbf{0.0127}$, $\mathbf{0.0233}$ and $\mathbf{0.0232}$ respectively, while mutual information obtained by PSOGSA is $\mathbf{0.0598}$.

Considering MSE (total Mean-Square quantization Error) as objective function for clustering algorithm we can mention Gravitational search Algorithm with K-means (GSA-KM) which built on the top of the K-means algorithm where the GSA help K-means to escape the local optima as K-means help GSA to converge fast by selecting the proper initial solution[22]. The algorithm outperformed other algorithm such as K-means, particle swarm, genetic algorithm, ant colony, honey bee matting and gravitational search algorithm.

Zhang et al. (2010) introduced Artificial Bee Colony algorithm (ABC) on the clustering problem and compared with GA, SA, TS, ACO and KM–PSO algorithm, results were very encouraging in term of solution quality and processing time [23].

In [24, 25] Tabu search has a trial in two forms, firstly pure Tabu search and secondly with the intervention of the K-means. The result was good in comparison with other single algorithms.

Chang et al. (2009) work on clustering with a genetic algorithm. The result was good also but suffering from low convergence speed in minority of the dataset [26].

Simulated annealing also introduced in [27] and the results were promising at that time.

Particle swarm optimization [28-31] contributed to this research area with different forms.

In [32] new algorithm based on one of the theories of the evolution of universe named Big Bang and Big Crunch theory(BB-BC) results in remarkable solution quality in all datasets even without combining the search with another meta-heuristic algorithm.

Two years later Black Hole (BH) algorithm was proposed that outperformed BB-BC and achieved the best result with least standard deviation in the majority of datasets [33].

Considering SSE (Sum of Squared Error) as objective function. Prakash et al. (2014) presented a comparison of evolutionary and swarm intelligence method for clustering. The comparison between algorithms shows that DE (Differential Evolution) and ABC (Artificial Bee Colony) outperform GA (Genetic Algorithm) and PSO (Particle Swarm Optimization [34].

Das et al. (2008) proposed an algorithm based on improved DE (Differential Evolution) which does not require a prior knowledge of the number of clusters, and comparing his algorithm with different types of clustering method that belongs to partitional and hierarchical, his algorithm tested in real datasets and show significant improvement over other algorithms in terms of inter-cluster as well as intra-cluster distance. DE algorithm run for $\mathbf{10^6}$ FE (Function Evaluation) which considered a long run time but with a desirable solution, DE algorithm also experimented on image segmentation and present promising results [35].

A simulated annealing algorithm also has some effort in clustering problem in [36]. The algorithm compared with the K-means algorithm in terms of solution quality, the initial temperature and cooling mechanism play key role for escaping the local optimum, the author concluded that as the number of iteration increased the solution quality is increased.

CHAPTER 4

PROPOSED ALGORITHM

In this chapter the proposed algorithm is thoroughly discussed. As an introduction we prefer to have a fast review on the Simulated Evolution Algorithm (SimE) upon which we base our proposed algorithm.

4.1 Simulated Evolution

Simulated Evolution algorithm (SimE) is a general search strategy for solving a variety of combinatorial optimization problem. It operates in a single solution. It begins from an initial solution, and keeps evolving the solution. The solution is becoming better from iteration to another. It assumes that there is a $\boldsymbol{P}$ of a set $\boldsymbol{M}$ of $\boldsymbol{N}$ (movable) elements; in addition there is a cost function. Cost that is used to compute the goodness $\boldsymbol{Gm}$ for each solution; SimE proceeds as following:

Initially the solution created randomly from the population while satisfying the problem constraint, SimE has the main loop which consists of three stages Evaluation, Selection and allocation. The stages are repeated by sequence till the solution goodness reaches it's maximum value or there is no noticeable improvement observed [37].

The non-deterministic behavior of the SimE enables the algorithm to escape local minima. SimE is a blind algorithm, similar to many other evolutionary algorithms. The blind algorithm should be told when to stop. Therefore, the runtime of SimE algorithm depends on the stopping criteria.

The stopping criteria and the size of the dataset together control the runtime of SimE algorithm [38]. The steps of the SimE algorithm are explained on next section

SimE Stages

4.1.1 Evaluation

This stage of evaluating the goodness of the solution where the goodness measure is a single number ranging from [0] to [1].

$$\boldsymbol{Gm} = \frac{\boldsymbol{Oi}}{\boldsymbol{Ci}} \tag{4.1}$$

In Equation (4.1) $\boldsymbol{Oi}$ is an estimation of the optimal cost and $\boldsymbol{Ci}$ is the current solution cost. $\boldsymbol{Oi}$ does not change from one iteration to another; the only changeable measure is $\boldsymbol{ci}$ which has to be minimized for increasing the overall goodness of the solution.

4.1.2 Selection

This stage is designed for selecting the elements which will be altered and moved from location, based on the goodness value for each element. The probability for the element to be altered in next generation is highly dependent on the goodness value of the element. Selection operator is non-deterministic where the high fitness elements still have a non-zero probability of being altered in next generation, which makes the algorithm capable of escaping the local minima.

4.1.3 Allocation

The allocation has most impact on the quality of the solution, it takes as input selected and non-selected sets and generates a new set $\boldsymbol{P'}$ containing members of previous set $\boldsymbol{P}$. Elements of selected set go for mutation based on a specified function of allocation.

The allocation function is a non-deterministic function that involves a selection among a set of possible mutation (moves) within the limit of upper and lower values; and there is a number of trials to be chosen based on the element fitness.

Finally the Allocation stage will favor the good move in most cases without turning the algorithm to a greedy one; FIGURE 4.1 below depicts the algorithm steps.

Algorithm Simulated Evolution (M,L)

M : set of movable elements

L: set of locations

B : selection bias

Initialization ;

Repeat

Evaluation :

For each element ∈ M Do $gm = \frac{Om}{Cm}$ **End;**

Selection :

For each element ∈ M Do

If selection (element,B) Then selected set =selected set ∪ {element}

Else non-selected set =non-selected set ∪ {element}

End If ;

Allocation :

For each member of selected set Do allocation (element)

End

Until stopping criteria met

Return (Best Solution)

End of algorithm

FIGURE 4.1: Simulated Evolution Stages.

4.2 SimE Applications

Simulated Evolution as approximation iterative heuristic is best suited to perform an intelligent search of the solution space. The algorithm works in a single and multiple objective optimization problems, SimE experimented on different applications such as VLSI standard cell placement [39], reservoir oil well placement problem [40], travelling salesman optimization problem [41] and obtained a desirable solutions.

4.3 Proposed SimE-based Algorithm

This chapter discusses the design of the proposed algorithm in detail.

Next section defines some key operations or terms used in the proposed algorithm.

4.3.1 Objective Function (MSE Based)

$$\boldsymbol{MSE} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{K}} \sum_{\forall\, \boldsymbol{Oi} \in \boldsymbol{Ci}} \boldsymbol{d}(\boldsymbol{Oi}, \boldsymbol{Ci})^{\boldsymbol{2}} \qquad (4.2)$$

Where the $\boldsymbol{d(Oi, Ci)}$ is the distance between the object $\boldsymbol{Oi}$ and cluster centroid Ci which the object assigned to, $\boldsymbol{K}$ is the number of clusters. The objective here is to minimize the distance measured by Euclidian distance and the equation below defines the Euclidian distance for two different objects Oi, Oj.

$$\boldsymbol{d(Oi, Oj)} = \sqrt{\left(\sum_{\boldsymbol{p=1}}^{\boldsymbol{l}} \left(\boldsymbol{O_i^p - O_j^p}\right)^{\boldsymbol{2}}\right)} \qquad (4.3)$$

$\forall\, l = total\ number\ of\ dimensions$

$\forall\, O^p = Attribute\ Value$

4.3.2 Movable Element

All algorithms under comparison are considered a cluster to be a unit of a movable element, this element evaluated by dissimilarity function as Equation (4.3), cluster cost equals to the sum of the dissimilarities with the objects assigned to it. The goal is to minimize the sum of dissimilarities.

In our algorithm, each attribute in the cluster centroid is a unit of the movable element and has to go through an evaluation step. On other words, the complexity of computations for evaluation stage in our algorithm is $\boldsymbol{O(K * l)}$ while for algorithms under comparison is $\boldsymbol{O(K)}$, SimE evaluation stage is performed for each attribute. For example if the dataset has a three clusters ($\boldsymbol{K = 3}$), then there are three possible movable elements for other algorithms, while in our proposed algorithm each attribute from the clusters centroids is a possible movable element, as example if the dataset have three clusters, $K = 3$ and the cluster itself has four dimensions $l = 4$ then there are $k * l$ possible movable elements which are equal to twelve, in this case, each possible movable element cost need to be calculated.

4.3.3 Proposed SimE-based Algorithm Design

This section describes the proposed design of the SimE algorithm. Each stage in the proposed SimE algorithm is described in the following.

4.3.3.1 Evaluation Stage

Cost function here is modified to adapt the change of the new type of movable elements (cluster attributes cost); in the proposed algorithm all attributes values are normalized in a range between [0, 1], then costs are calculated after assignments of the objects to the closest cluster based on distance measure from Equation (4.3).

$$\boldsymbol{attr(m)cost} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{s}} \left(\mathbf{O}_{\boldsymbol{i}}^{\boldsymbol{m}} - \mathbf{C}^{\boldsymbol{m}}\right)^{\mathbf{2}}} \tag{4.4}$$

$$\boldsymbol{Attribute\ fitness} = \mathbf{1} - \boldsymbol{attr(m)cost} \tag{4.5}$$

Equation (4.4) is to calculate the cost of attribute $\boldsymbol{m}$ in the cluster centroids. Where Equation 4.5 calculates the attribute fitness.

$\forall\ s = number\ of\ the\ objects\ assigned\ to\ \ cluster$

$\forall\ m = attribue\ order$ from $(1\ to\ l)$

$\forall\ O = objects\ assigned\ to\ cluster\ C$

Cost calculated $(k * l)$ times to build the evaluation matrix for all elements, TABLES 4.1 and 4.2 highlight the concept.

As an example of the mentioned statement above let us consider TABLE 4.1 below which has K= 3 centroids and each centroid with four dimensions.

Attributes / Centroids	Attr1	Attr2	Attr3	Attr4
C1	**0.1**	0.3	1	0.34
C2	0.2	0.2	0.5	0.78
C3	0.12	0.33	0.44	.19

TABLE 4.1: Centroids Dimensions Representation.

Attributes / Objects	Attr1	Attr2	Attr3	Attr4
Object 1	**0.7**	0.5	0.33	0.61
Object 2	**0.9**	0.13	0.93	1
Object 3	1	0.24	0.44	0.93
Object 4	0.7	0.5	0.33	0.61

TABLE 4.2: Objects Representation with Four Attributes.

Assume Obj1 and Obj2 are assigned to C1, using Equation 4.4 to calculate the cost of the 1st attribute for C1.

$$\boldsymbol{Attribute\ cost\ (1)} = \sqrt{(\mathbf{0.7}-\mathbf{0.1})^2+(\mathbf{0.9}-\mathbf{0.1})^2} =$$

$$\sqrt{(\mathbf{0.49}-\mathbf{0.01})^2+(\mathbf{0.81}-\mathbf{0.01})^2} = \mathbf{0.8704}$$

The attribute cost subtracted from 1 to transfer the cost value into fitness value where the higher number indicate better attribute value and less probability to be selected for allocation (mutation).

$$\text{Attribute fitness}(1) = 1 - 0.8704 = 0.129$$

4.3.3.2 Selection Stage

Selection is the second stage where attributes selected based on acceptance probability condition as below;

$$\boldsymbol{if\ Random >= fitness\ value - \beta} \tag{4.6}$$

$\forall\ \boldsymbol{Random}\ is\ uniformly\ distributed\ random\ number\ in\ the\ interval(\mathbf{0}, \mathbf{1})$

$\forall\ \boldsymbol{\beta}\ \ is\ a\ value\ between\ [0, 0.2]$

In most cases $\boldsymbol{\beta}$ value is selected to be $(\mathbf{0.1})$, if solution goodness is relatively far from the best solution $\boldsymbol{\beta}$ has to be greater than $(\mathbf{0.1})$ to extremely explore different areas by selecting more elements in the selection group. The Attributes which satisfy condition 4.6 are selected for allocation stage.

4.3.3.3 Allocation Stage

Allocation is the last stage of the algorithm. Where attributes in selection group have to be allocated or unchanged at the end of this stage, the defined conditions for the attributes which have to be allocated to the new value or kept unchanged explained by pseudo code of allocation stage in FIGURE 4.2.

Π % bad move acceptance probability

$\mathcal{E}$ % accepted solution quality drop

£ % Attribute disturbance value

1: for i=1 to S % S is total number of members in selection group

2: for j =1: to W % W is window of trials

3: Generate $\boldsymbol{\sigma}$ % $\boldsymbol{\sigma = random\ value\ between\ (-1, 1)}$

4: $\boldsymbol{ATR_{ij}} = \boldsymbol{ATR_i} + \boldsymbol{\sigma} * £$ % compute the new attribute

5: End j loop

6: Generate $\mathbf{R}$ % $\boldsymbol{R\ is\ a\ random\ value\ between\ (0, 1)}$

7: Compute the fitness value of all the new W trials attributes

---

8: If new fitness > original fitness then

9: ATR= $\boldsymbol{ATR_{ij}}$

10: Else if $\boldsymbol{ATR_{ij}}$ **fitness** $< \boldsymbol{original\ fitness}$ and $(\boldsymbol{R} \leq \boldsymbol{\Pi})$ and

$$(\mathbf{100} * \frac{\boldsymbol{old\ fitness - new\ fitness}}{\boldsymbol{old\ fitness + new\ fitness}}) \leq \boldsymbol{\mathcal{E}}$$

11: ATR= $\boldsymbol{ATR_{ij}}$

12: Else keep the attribute unchanged

13: End if

14: End i loop

FIGURE 4.2: Pseudo Code for Allocation Stage.

Allocation stage receives attributes and their fitness value as input. For each attribute there are $\boldsymbol{W}$ trials (trial window) to change the attribute value, the original attribute disturbed by value on the range $\pm(\mathbf{0},\mathbf{0.02})$ as stated on the Equation 4.7 below.

$$\boldsymbol{ATR_{ij}} = \boldsymbol{ATR_{i}} + \boldsymbol{\sigma} * £ \tag{4.7}$$

$\forall \quad \boldsymbol{\sigma} = random\ value\ between\ (-1,1)$

$\forall \quad £ = 0.02$

The previous equation applied $\boldsymbol{W}$ times and have output of $\boldsymbol{W}$ offspring's attributes, next to that each attribute fitness value calculated based on attribute fitness equation.

The new attribute acceptance cases will be explained in detail below.

Case [1]

If all $\boldsymbol{W}$ new attributes have better fitness than the original one then we select the best attribute fitness among the $\boldsymbol{W}$ attributes to replace the original one.

Case [2]

If all W new attributes have fitness worse than the original one, then there is a probability of accepting the bad move only if the below two conditions are satisfied.

1. R ≤ Π

   ∀ R $is\ a\ random\ value\ between\ \ (0,1)$

   ∀ $\Pi\ is\ a\ bad\ move\ acceptance\ probability$

2. $\left(100 * \frac{old\ fitness - new\ fitness}{old\ fitness + new\ fitness}\right) \leq \varepsilon$

   ∀ $\varepsilon\ Maximum\ solution\ quality\ drop$

ε-Value set to 10 percentages drops in solution quality to control the solution quality drop during the search.

Case [3]

If all W new attribute does not satisfy Case 1 or 2 conditions then keep the attribute unchanged. FIGURE 4.3 depict algorithm flow chart.

Proposed algorithm classified as a partitional method which starts from random K partitions (centroids) or it can start from the K-means solution as initial state as it will be explained on the experiment chapter, then we compute the fitness for each attribute on the centroid by Equation 4.5 and keep changing the centroids values towards better solution quality.

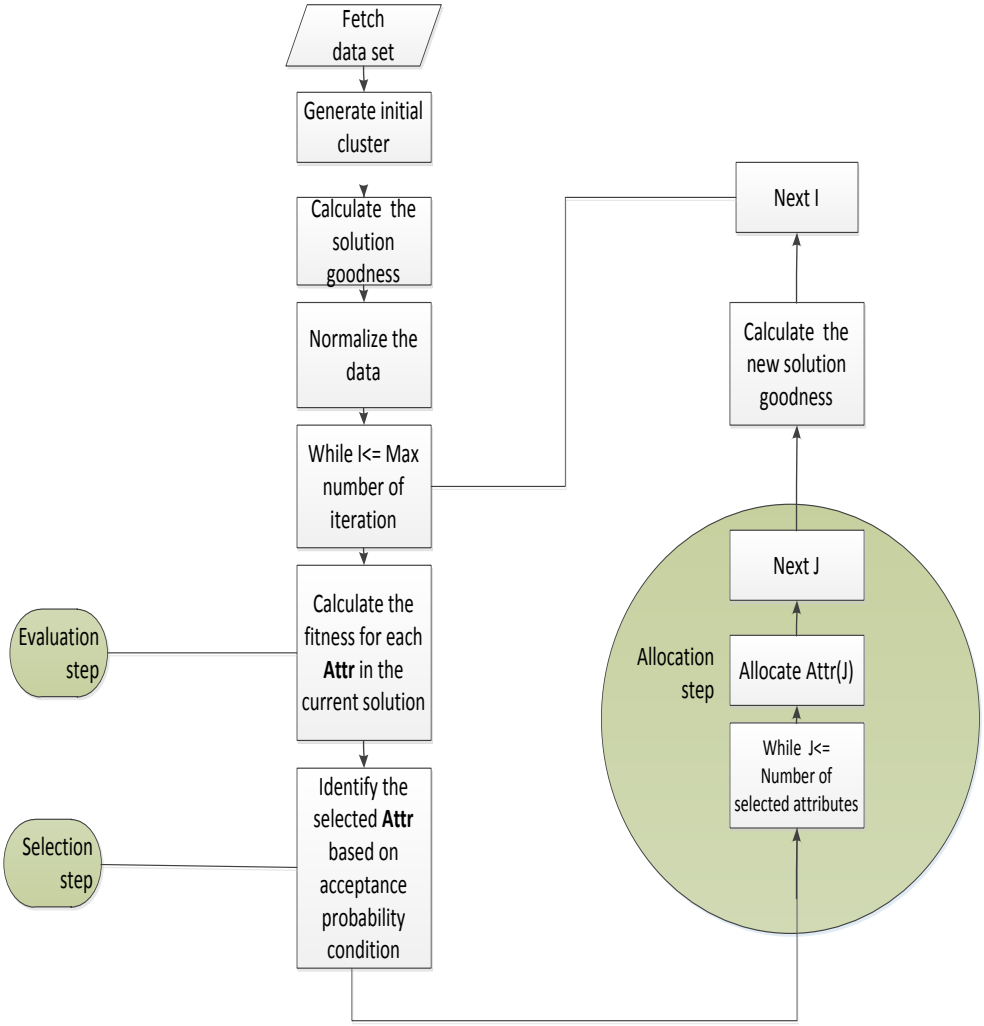

FIGURE 4.3 : Flow Chart of the Proposed Algorithm.

4.3.4 Mutation Operation

The mutation operation in here is representing the allocation stage in SimE Algorithm. Selection set (Ps) is a set of attributes which are selected to go for mutation stage because of their low fitness value, with a small probability the selection set can contain an attribute with high fitness value.

Mutation condition and design are explained in below steps:

1. Each attribute under mutation will have $\boldsymbol{w}$ trials. By adding a value on the range $\pm(\mathbf{0},\mathbf{0}.\mathbf{02})$ to the attribute value.
2. The best trial out of these trials are selected to be the new attribute if it's fitness is more than the original attribute under mutation.
3. If both trials fitness are not better than the original attribute the bad move is accepted based on the acceptance function.
4. Acceptance function accepts the bad move by a probability of $(\boldsymbol{\Pi})$ if the fitness decreased in a percentage not more than ten percent.
5. The remaining (1- $\boldsymbol{\Pi}$) probability is to keep the original attribute unchanged.

4.3.5 Complete Execution Flow of the Proposed Algorithm

1. Fetching the dataset under study.

2. Normalizing the dataset.

3. Initiating a random solution which has the centroid of the K cluster.

4. Starting the loop.

5. Assigning each data point to the closest centroid based on the Euclidean distance measure.

6. Computing the fitness for each attribute of the clusters $(\boldsymbol{C})$ on the current assignment using Equation 4.5.

7. Building matrix of attributes fitness.

8. Identifying the selection set $(\boldsymbol{Ps})$ which will go for mutation based on some probability.

9. Starting mutation operation loop for each attribute on the selection set$(\boldsymbol{Ps})$, with a window of $(\boldsymbol{W})$**.**

10. For each attribute under mutation If the new attribute fitness is maximized then accepts the move (good move) else if the new attribute fitness minimized accepts the bad move with some probability (acceptance function) or keep the attribute with no change.

11. After completing the mutation operation loop for all the selected attributes, reformulating the new centroids.

12. Reassigning the objects to the closest new centroids using distance measure as Equation 4.3.

13. De-normalizing centroids values.

14. Calculating the current solution quality using Equation 4.2 for MSE.

15. Saving the best solution centroids attribute.

16. Updating the values of the best solution If current solution is better than previous.

17. Normalizing current iteration solution.

18. Stopping the code If the number of iteration hit the maximum number.

19. Else go to step 4 and feed the new iteration with current solution normalized value.

CHAPTER 5

EXPERIMENTAL RESULTS

This chapter discusses experiments environment, parameters configurations and results.

5.1 Dataset Specification

Five real datasets validate our algorithm. Datasets are iris, Wine, Glass, Breast Cancer Wisconsin (Cancer), and Contraceptive Method Choice (CMC) [42]. Having a predefined number of clusters, TABLE 6.1 gives the datasets details. These data sets are used by different authors in literature to evaluate and validate the results.

IRIS: This dataset contains three categories of fifty objects each, and each category refers to a type of iris flower, the data set contains four numeric attribute sepal length, and width in cm, petal length, and width in cm.

WINE: This datasets contains the chemical results analysis of wine grown in the same region in Italy derived from three different cultivars. The dataset contains thirteen continuous numeric attributes.

GLASS: This dataset contains a sample from six different types of glass, the data sets contains nine numeric attributes

Contraceptive Method choice CMC: This is a subset of 1987 national Indonesian contraceptive prevalence survey, the samples are married women who either were not pregnant or do not know if they were on the time of the diverse type. The problem is to predict the choice of current contraceptive method

Breast cancer Wisconsin: This dataset contains nine attributes clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses.

Dataset	Number of clusters	Number of attributes	Number of data objects
Iris	3	4	150
Wine	3	13	178
Glass	6	9	214
Breast cancer	2	9	683
CMC	3	10	1473
CMC	3	9	

TABLE 5.1: Datasets Specification.

5.2 Discussion of Experiments Results

The algorithm has been implemented in Matlab 7.0.4 on T3400, 2.16GHZ, 4GB RAM computer. The K-means algorithm implemented by Matlab function considering the experiment modifications.

We have compared the performance of the proposed SimE based Algorithm with Big Bang–Big Crunch (BB-BC) [32], Black-hole (BH) [33], K-means and a random version of the proposed algorithm itself.

Proposed algorithm is also incorporating the K-means as initial solution (SimE-KM) and compared with Gravitational Search Algorithm with K-means (GSA-KM), Gravitational Search Algorithm (GSA), Genetic Algorithm (GA), Simulated Annealing (SA), Ant colony Optimization (ACO), Honey Bee Mating Optimization (HBMO), and Particle Swarm Optimization (PSO).

5.2.1 First Experiment

First experiment: We compared our algorithm with BB-BC and BH. Solution quality was evaluated based on (MSE) measure on the five datasets. Experiment is conducted for 20 independent runs to avoid randomized nature of the solution. The minimum the value is the better solution. MSE average, minimum and the standard deviation of the 20

independent runs are calculated and recorded on TABLE 5.2. CMC dataset is tested here with two configuration sets. Firstly, when the number of $Attributes = 9$ to compare with BB-BC and BH results. Secondly, when the number of $Attributes = 10$ to compare with BB-BC results.

As we can see in TABLE 5.2 below, the proposed algorithm outperforms BH and BB-BC algorithms in terms of average, best and standard deviation of MSE value for IRIS, CMC (both configurations), Wine and breast cancer datasets. For Glass dataset our algorithms obtain minimum MSE value among other algorithms. It is also obtained better average and standard deviation than BB-BC but not better than the BH. The lowest value of standard deviation explains the ability of our algorithm to find global optimum solution in most runs on the datasets.

TABLES 5.3 through 5.8 show best cluster centroids obtained for all datasets, tables can be helpful for future works to validate the accuracy of MSE objective function by plugin the best centroid from our tables to the researchers' objective function and matching the output value with best MSE value obtained in TABLE 5.2.

Dataset	MSE Value	BB-BC [32]	BH[33]	SimE
IRIS	Best	96.67718	96.65589	**96.6553**
	Average	96.77319	96.65681	**96.6556**
	Std	0.2226	0.00173	**0.0002**
	NFE	-	-	4.47E+05
WINE	Best	16299.53193	16,293.42	**16292**
	Average	16304.29787	16,294.32	**16292.88**
	Std	2.87718	1.65127	**0.7**
	NFE	-	-	4.95E+05
Glass	Best	223.8941	210.51549	**210.0020**
	Average	231.23058	211.4986	211.905
	Std	4.65013	1.1823	2.136
	NFE	-	-	4.96E+05
CMC(9)	Best	5534.09483	5532.88323	**5532.2**
	Average	5574.75174	5533.63122	**5532.3**
	Std	39.43494	0.5994	**0.09**
	NFE	-	-	4.32E+05
CMC(10)	Best	5700.63051		**5693.7**
	Average	5744.03239		**5693.721**
	Std	26.63582		**0.0077**
	NFE	-	-	4.92E+05
Cancer	Best	2964.38764	2964.38878	**2964.3870**
	Average	2964.38813	2964.39539	**2964.3876**
	Std	0.0005	0.00921	**0.0003**
	NFE	-	-	4.97E+05

TABLE 5.2: First Experiment Simulation Results for Clustering Algorithms.

C1	C2	C3
5.012	5.9341	6.7334
3.4031	2.797	3.0677
1.4715	4.4175	5.63
0.23546	1.4178	2.1059

TABLE 5.3: The Best Centroids Obtained on the IRIS Dataset.

C1	C2	C3
12.513	12.82	13.759
2.3646	2.5352	1.8651
2.3193	2.3858	2.4314
21.305	19.514	16.92
92.52	98.95	105.27
2.0451	2.076	2.8573
1.7792	1.4945	3.0617
0.41052	0.41877	0.28969
1.4365	1.4232	1.9589
4.3518	5.7647	5.6911
0.95135	0.88554	1.0857
2.4643	2.2138	3.0326
463.59	686.97	1137.3

TABLE 5.4: The Best Centroids Obtained on the Wine Dataset.

C1	C2
7.1171	2.8893
6.6416	1.1280
6.6285	1.2014
5.6146	1.1643
5.2410	1.9935
8.1007	1.1217
6.0782	2.0054
6.0221	1.1013
2.3268	1.0316

TABLE 5.5: The Best Centroids Obtained on the Cancer Dataset.

C1	C2	C3	C4	C5	C6
1.5172	1.5165	1.5214	1.5174	1.5216	1.5131
13.317	14.635	13.802	12.845	13.101	13.01
3.5872	0.0643	3.5534	3.4595	0.25001	0
1.4233	2.2126	0.93603	1.3066	1.4274	3.0299
72.671	73.268	71.856	73.015	72.684	70.591
0.57657	0.048136	0.16707	0.58738	0.30125	6.21
8.2016	8.6938	9.5246	8.5692	11.975	6.9447
0.010269	1.0052	0.03483	0.00765	0.050252	0
0.038363	0.018669	0.053298	0.074138	0.058125	0

TABLE 5.6: The Best Centroids Obtained on the Glass Dataset.

C1	C2	C3
33.507	24.413	43.648
3.1319	3.039	2.9891
3.5505	3.5096	3.4455
3.6632	1.7898	4.5911
0.78928	0.92693	0.79705
0.69539	0.79071	0.76582
2.1025	2.2949	1.8351
3.2852	2.9724	3.423
0.060501	0.03639	0.093686
2.1146	2.001	1.6673

TABLE 5.7: The Best Centroids Obtained on the CMC with Ten Attributes Dataset.

C1	C2	C3
24.416	43.638	33.495
3.0447	2.9965	3.1296
3.5123	3.4538	3.5531
1.7898	4.5862	3.6504
0.92534	0.79577	0.79431
0.79077	0.76403	0.69442
2.3028	1.8257	2.0996
2.9725	3.4302	3.2822
0.036666	0.090952	0.06004

TABLE 5.8: The Best Centroids Obtained on the CMC with Nine Attributes Dataset.

5.2.1.1 Parameter Configuration for the First Experiment

As discussed on the previous section the first experiment compared the proposed algorithm with algorithms that obtained the best results of MSE value in recent studies; even though runtime was not a measured criteria on their study, we had recorded the runtime for future works.

Another measure is added to the experiment which is the number of function evaluations (NFE) that indicates the convergence speed of the respective algorithm. NFE is the number of times that the clustering algorithm has calculated the objective function Eq. 4.2 to reach the near optimal solution. It is dependent on the number of iterations to reach the optimal solution.

Parameters for this experiment are fixed for all datasets. The experiment initialized from a random solution, and then iterations start improving the solution quality till it reaches the stopping criterion which is the run time on this experiment. TABLE 5.9 details the maximum runtime for each dataset and parameters configuration.

By running the algorithm many times with different setups we found that the best value of mutation trials window($\boldsymbol{w}$) is to be set to the value of two. $\boldsymbol{\beta}$ Value on the selection stage is configured to $\mathbf{0.1}$ as the standard SimE algorithm; in addition to that the bad move acceptance probability $\boldsymbol{\Pi}$ is having a value of 0.05.

TABLE 5.10 displays the number of times fitness function is calculated to obtain the best result for each dataset. This value can be calculated as below:

$$No\ of\ fitness\ function\ =$$

$$NFE * total\ no\ of\ ttributes + \left(w * \sum_{i=1}^{Bestiter} sitem(i)\right) \quad (5.1)$$

$\forall\, l$ w = the number of trials for allocation stage

$\forall\, l$ $Bestiter$ = the number of iterations required to reach the best solution

$\forall\, l$ $sitem(i)$= the number of items on selection set

Dataset	Runtime (min)	No of iterations	w	β	Π
IRIS	30	$5 * 10^5$	2	0.1	0.05
Wine	40				
Glass	70				
CMC(9)	200				
CMC(10)	200				
Breast Cancer	75				

TABLE 5.9: First Experiment Configurations.

Dataset	Number of fitness function
IRIS	7.57E+06
Wine	2.19E+07
Glass	3.29E+07
CMC(9)	1.52E+07
CMC(10)	1.86E+07
Breast Cancer	1.28E+07

Table 5.10: Number of Fitness Function for the Best Run in First Experiment.

5.2.2 Second Experiment

On the Second experiment we compared our algorithm with K-means and random search algorithm. Solution quality for the five datasets evaluated based on (MSE) measure.

Random search algorithm is a random version of the proposed algorithm where the selection and allocation stage is randomized. The comparison with random search is conducted to negate the randomness nature of our proposed algorithm operations.

Implemented K-means shows that K-mean solution is not improving after the first 100 iterations; hence to run K-means in a fair condition with other algorithms that have their solution quality improved iteration after another, we increase the number of initial solutions and decrease the number of iterations per run for the K-means experiment.

Proposed algorithm and random search are running with the same setup as on the first experiment. For K-means it is initialized from a random solution and it runs for 100 iterations before turning to another starting point. This process repeats itself till it reaches the stopping criterion (run time). The MSE value at the end of each 100 iterations is recorded and the best one through the run is selected. Parameters values for this experiment are fixed on the proposed algorithm and random search algorithm for all datasets as shown on TABLE 5.9 on the previous section. TABLE 5.11 shows the results for the second experiment.

Dataset	Run time in minutes	MSE Value	Random Search	K-means	SimE
IRIS	30	**Best**	101.5958	97.3259	**96.6556**
		Average	106.6808	101.2479	**96.6553**
WINE	40	**Best**	34165.000	16878.0000	**16292**
		Average	40349.150	16556.0000	**16292.88**
Glass	70	**Best**	521.6025	215.4704	**210.0020**
		Average	580.362	220.0709	**211.905**
CMC(9)	75	**Best**	8182.4120	5543.5000	**5532.2**
		Average	8581.4605	5543.9450	**5532.3**
CMC(10)	200	**Best**	7594.0410	5703.2000	**5693.7**
		Average	9019.0351	5704.1900	**5693.721**
Cancer	200	**Best**	2980.8000	2988.4000	**2964.3870**
		Average	2986.9150	2988.4000	**2964.3876**

TABLE 5.11: Second Experiment Simulation Results for Clustering Algorithms.

As shown on TABLE 5.11 results confirmed that our proposed algorithm is performing better than K-means and random algorithm in all benchmarks; moreover, our proposed algorithm solution quality is very far from the random algorithm. This negates randomness hypothesis for the proposed algorithm.

5.2.2.1 Statistical Significance of the Results

In principle, a statistically significant result (usually a difference) is a result that's not attributed to chance. More technically, it means that if the Null Hypothesis is true (which means there really is no difference), there's a low probability of getting a result which is large or larger [43].

In order to find significant differences among the results obtained by the K-means and the proposed algorithm, statistical analysis is carried out. T-test for all datasets is performed in two tails for SimE vs. K-mean with significance level (0.05). Our results are statistically significant in all the datasets except IRIS as can be shown on the TABLE below.

Dataset	P(T<=t) one-tail	P(T<=t) two-tail
iris	**0.0421413**	0.084282548
Wine	**0.0004288**	**0.000857549**
Glass	**0.0000864**	**0.000172764**
CMC (9)	**5.73415E-25**	**1.14683E-24**
CMC (10)	**1.91885E-20**	**3.8377E-20**
Cancer	**1.33784E-69**	**2.67568E-69**

TABLE 5.12: T-TEST for K-means Vs. Proposed Algorithm.

5.2.3 Third Experiment

On the third experiment we implemented the new version of the proposed algorithm Simulated Evolution with K-means (SimE-Km).The proposed algorithm takes the advantage of K-means to initialize the algorithm from a high quality solution which speeds up the convergence time.

SimE-Km is compared with eight algorithms, namely: K-means, Genetic Algorithm (GA), Simulated Annealing (SA), Ant Colony Optimization (ACO), Honey Bee Mating Optimization (HBMO), Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA) and Gravitational Search Algorithm with K-means (GSA-KM).

Twenty independent runs were conducted for each experiment. Average, best and worst MSE value achieved is recorded. The NFE value is also recorded as second measure.

SimE-Km experiments were conducted with two configuration based on the maximum number of NFE as we will explain in Sections 5.2.3.1 and 5.2.3.2.

Algorithm parameters for the two configurations are fixed; TABLE 5.13 details the parameter values and maximum allowed NFE for both configurations.

Dataset	*Max NFE Conf(1)*	*Max NFE Conf(2)*	*Number of movable elements/iteration*	*w*	*β*	*Π*
IRIS	*38 128*	*4628*	*4*	*1*	*0.5*	*0.5*
Wine	*33 551*	*7238*				
Glass	*199 892*	*171 910*				
CMC	*29 483*	*11 796*				
Breast Cancer	*20 221*	*8262*				

TABLE 5.13: Third Experiment Configurations.

The number of movable elements per iteration is fixed to 4 elements and the number of trials per iteration ($\boldsymbol{w}$) is fixed to value of 1.

$\boldsymbol{\beta}$ Value on the selection stage is configured to 0.5 value to help the algorithm in performing intense search; in addition to that the bad move acceptance probability $\boldsymbol{\Pi}$ is increased to the value of 0.5 which enables the proposed algorithm for escaping the local optima nature of initial solution provided by K-mean.

5.2.3.1 Third Experiment (First Configuration)

SimE-Km on the first configuration has been limited to the maximum NFE obtained by slowest algorithm in comparison which belongs to the genetic algorithm. TABLE 5.14 summarizes the results in comparison with other algorithms.

The results explain the ability of the proposed hybrid algorithm to obtain the best MSE value compared to the competitors' algorithms in all datasets. The proposed algorithm obtained the best average from the 20 independent runs for all datasets.

SimE-Km in terms of the worst solution has achieved best value in all datasets except on the glass dataset, where GSA-KM worst MSE value was $\mathbf{216.08}$ when SimE-Km worst value was $\mathbf{216.445}$.

Even though GSA-Km NFE is smaller than SimE-Km but the solution quality obtained by the SimE-Km is not achieved by GSA-KM in any single run out of the 20 runs for all datasets. The NFE value for SimE-KM is four times the NFE value of GSA-KM in most cases.

dataset	Criteria	*K-means*	*GA*	*SA*	*ACO*	*HBMO*	*PSO*	*GSA*	*GSA-KM*	*SIME-Km*
	Best	97.333	113.98	97.45	97.10	96.75	96.894	96.698	96.679	**96.6556**
IRIS	Average	106.050	125.19	99.95	97.17	96.95	97.232	96.723	96.689	**96.6560**
	Worst	120.450	139.77	102.01	97.80	97.75	97.897	96.764	96.705	**96.6582**
	NFE	120	38 128	5314	10 998	11 214	4953	4628	**1377**	8688
	Best	16 555.68	16 530.53	16 473.48	16 530.53	16 357.28	16 345.96	16 315.35	16 294.25	**16293**
Wine	Average	18 061.00	16 530.53	17 521.09	16 530.53	16 357.28	16 417.47	16 376.61	16 294.31	**16294.1**
	Worst	18 563.12	16 530.53	18 083.25	16 530.53	16 357.28	16 562.31	16 425.58	**16 294.64**	16295
	NFE	390	33 551	17 264	15 473	7238	16 532	15 300	**5523**	7690
	Best	215.74	278.37	275.16	269.72	245.73	270.57	220.78	211.47	**210.4494**
Glass	Average	235.50	282.32	282.19	273.46	247.71	275.71	225.70	214.22	**212.9402**
	Worst	255.38	286.77	287.18	280.08	249.54	283.52	229.45	216.08	216.4450
	NFE	630	199 892	199 438	196 581	195 439	198 765	171 910	**1759**	48709
	Best	5842.20	5705.63	5849.03	5701.92	5699.26	5700.98	5698.15	5697.03	**5693.8**
CMC	Average	5893.60	5756.59	5893.48	5819.13	5713.98	5820.96	5699.84	5697.36	**5694.255**
	Worst	5934.43	5812.64	5966.94	5912.43	5725.35	5923.24	5702.09	5697.87	**5695.8**
	NFE	270	29 483	26 829	20 436	19 496	21 456	11 796	**1754**	7175
	Best	2999.19	2999.32	2993.45	2970.49	2989.94	2973.5	2967.96	2965.14	**2964.4**
	Average	3251.21	3249.46	3239.17	3046.06	3112.42	3050.04	2973.58	2965.21	**2964.5**
Cancer	Worst	3521.59	3427.43	3421.95	3242.01	3210.78	3318.88	2990.83	2965.30	**2964.6**
	NFE	180	20 221	17 387	15 983	19 982	16 290	8262	**1642**	4852

TABLE 5.14: Third Experiment Simulation Results for Clustering Algorithms (1st configuration).

TABLE 5.15 displays the number of times fitness function is calculated to obtain the best result for each dataset.

Dataset	Number of fitness function
IRIS	1.39E+05
Wine	3.31E+05
Glass	2.83E+06
CMC	2.44E+05
Breast Cancer	1.07E+05

TABLE 5.15 : Number of Fitness Function for the Best Run in Third Experiment (1st Configuration).

TABLES 5.14 and 5.15 show NFE and number of fitness function for the hybrid algorithm (SimE-Km); these values are significantly reduced in comparison with starting from a random solution as shown on TABLES 5.2 and 5.10.

5.2.3.2 Third Experiment (Second Configuration)

SimE-Km on the second configuration has been compared to a subset of the previous algorithms. Those algorithms are Genetic algorithm, Simulated Annealing, Ant Colony Optimization, Honey Bee Mating Optimization, particle Swarm Optimization and Gravitational Search

Algorithm. The maximum NFE on this experiment is limited to the fastest algorithms in each dataset. TABLE 5.16 summarizes the results.

dataset	Criteria	*GA*	*SA*	*ACO*	*HBMO*	*PSO*	*GSA*	*SIME-Km*
IRIS	Best	113.98	97.45	97.10	96.75	96.894	96.698	**96.6637**
	Average	125.19	99.95	97.17	96.95	97.232	96.723	**96.7051**
	Worst	139.77	102.01	97.80	97.75	97.897	96.764	**97.2210**
	NFE	38 128	5314	10 998	11 214	4953	4628	**1067**
Wine	Best	16 530.53	16 473.48	16 530.53	16 357.28	16 345.96	16 315.35	**1.6297**
	Average	16 530.53	17 521.09	16 530.53	16 357.28	16 417.47	16 376.61	**16302.4500**
	Worst	16 530.53	18 083.25	16 530.53	16 357.28	16 562.31	16 425.58	**16321.0000**
	NFE	33 551	17 264	15 473	7238	16 532	15 300	**1564**
Glass	Best	278.37	275.16	269.72	245.73	270.57	220.78	**210.4491**
	Average	282.32	282.19	273.46	247.71	275.71	225.70	**215.4552**
	Worst	286.77	287.18	280.08	249.54	283.52	229.45	**222.4496**
	NFE	199 892	199 438	196 581	195 439	198 765	171 910	**41985**
CMC	Best	5705.63	5849.03	5701.92	5699.26	5700.98	5698.15	**5694.2000**
	Average	5756.59	5893.48	5819.13	5713.98	5820.96	5699.84	**5695.3650**
	Worst	5812.64	5966.94	5912.43	5725.35	5923.24	5702.09	**5697.0000**
	NFE	29 483	26 829	20 436	19 496	21 456	11 796	**2926**
Cancer	Best	2999.32	2993.45	2970.49	2989.94	2973.50	2967.96	**2964.6000**
	Average	3249.46	3239.17	3046.06	3112.42	3050.04	2973.58	**2965.0200**
	Worst	3427.43	3421.95	3242.01	3210.78	3318.88	2990.83	**2966.7000**
	NFE	20 221	17 387	15 983	19 982	16 290	8262	**1980**

TABLE 5.16: Third Experiment Simulation Results for Clustering Algorithms (2nd configuration).

The main objective on this experiment is to confirm the fast convergence of the proposed algorithm in comparison with all algorithms excluding K-means and GSA-Km.

The experiment shows that the SimE-Km outperforms those algorithms on the five datasets for all quality measures MSE average, best and worst in addition to the NFE. Results show that the proposed algorithm has achieved best solution quality with less than ¼ NFE values of the fastest algorithms under comparison.

The number of times fitness function is calculated for each dataset is shown in TABLE 5.17 below.

Dataset	Number of fitness function
IRIS	1.71E+04
Wine	6.73E+04
Glass	2.44E+06
CMC	9.95E+04
Breast Cancer	4.36E+04

TABLE 5.17: Number of Fitness Function for the Best Run in Third Experiment (2nd Configuration).

FIGURES 5.1 and 5.2 plot the reciprocal value of MSE versus the number of iteration. FIGURE 5.1 shows the solution quality improvement over 1000 iterations in a single run of SimE-Km for the IRIS dataset. While FIGURE 5.2 shows the solution quality improvement over 1800 iterations in a single run of SimE-Km for the Wine dataset. The reader can observe

that SimE-Km achieved ¾ of the whole improvement when it reaches half of the total number of iterations. The algorithm keeps on improving till it reaches the last 10 percent of iterations when the improvement is non-noticeable.

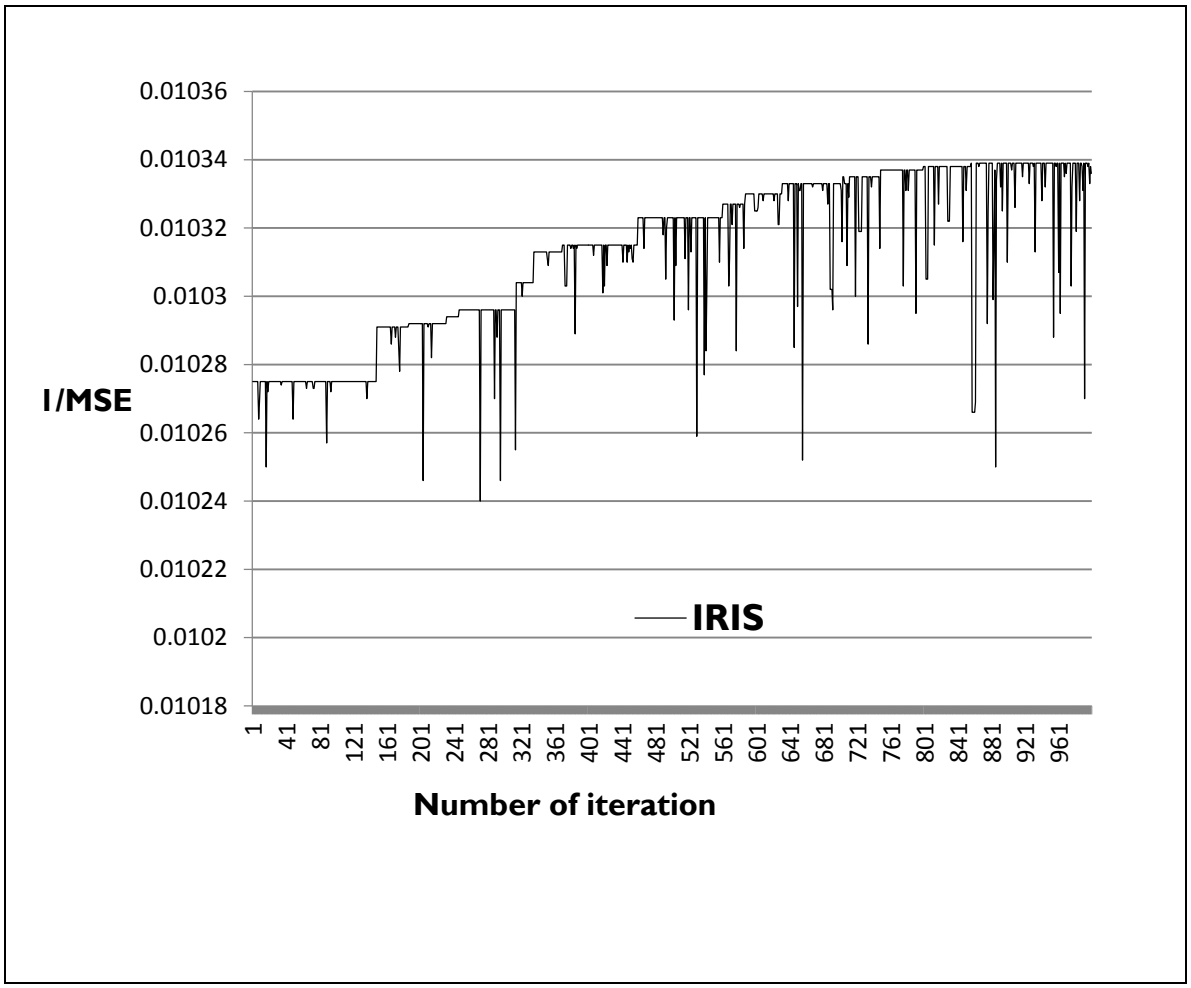

FIGURE 5.1: Solution Quality Improvement over Iterations for IRIS Dataset.

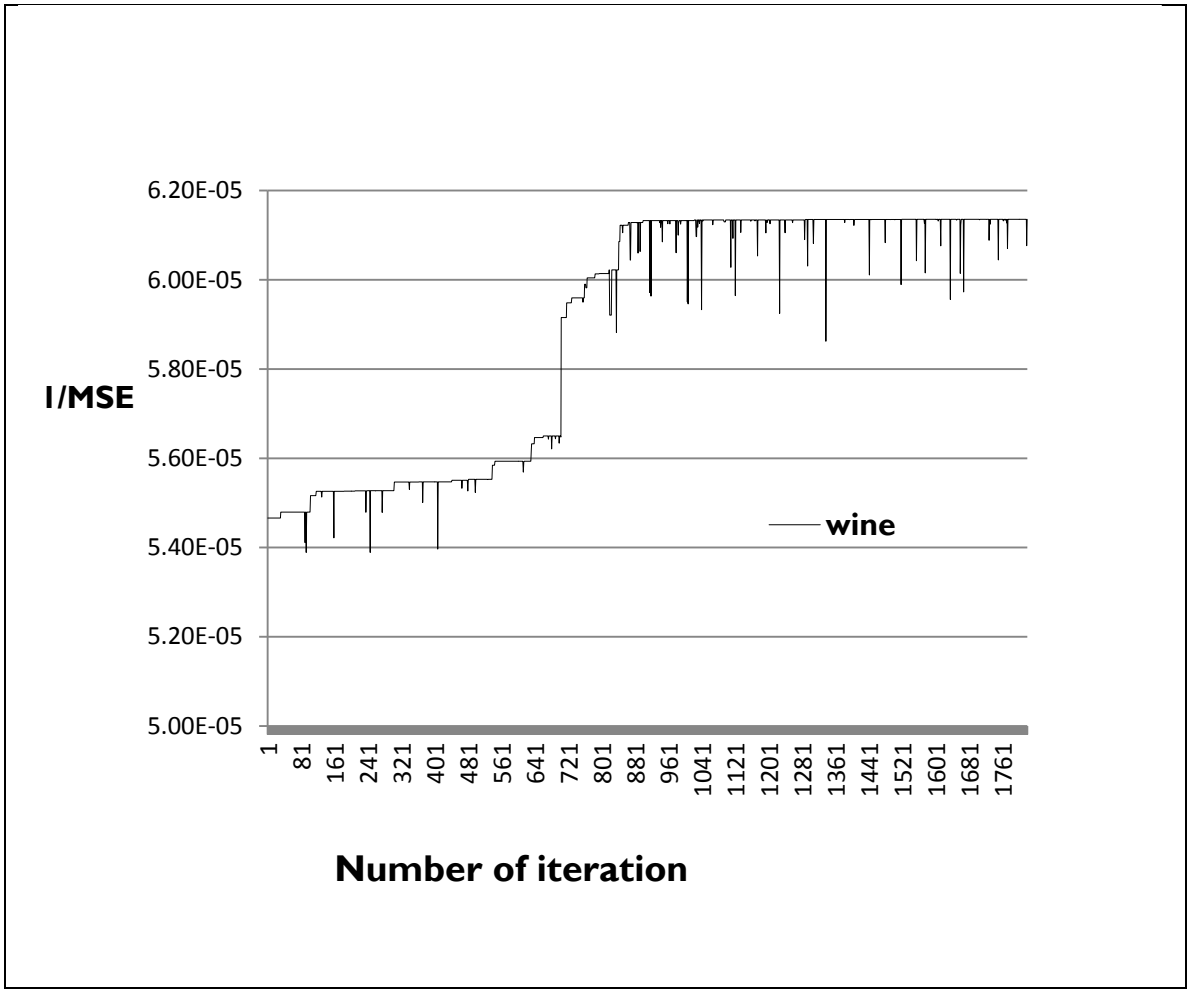

FIGURE 5.2: Solution Quality Improvement over Iterations for Wine Dataset.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, a SimE based algorithm has been proposed to solve the data clustering problem. We used the Mean Square quantization Error (MSE) as goodness measure for its wide area of applications and recent research related to it. The allocation operation targeted the centroids attributes as a unit of movable element .The solution quality was computed using MSE validity index. The results showed that the solution quality of the proposed algorithm is competitive to the existing ones. Multiple trials of the proposed algorithm showed very small standard deviation. In Future, the proposed algorithm can be improved to obtain the best solution quality for huge datasets by tuning algorithm operators. Currently, the runtime of the proposed algorithm requires some improvement; however, parallelization can be used to improve the runtime. Another area of future work is to adapt the proposed algorithm in practical applications of clustering such as quantization.

Appendix

Matlab code:

```
**********************************************************************************************
clear all

%************************ DISPLAY BOX TO SELECT DATASE*****************%
textI=('This code is implemented to run an algorithm for clustering a dataset which is selected by the user ');
text2=('1.IRIS');
text3=('2.Wine');
text4=('3.Breast cancer');
text5=('4.CMC');
text6=('5.Glass');
text7=('6.CMC1');
disp(textI);
disp(text2);
disp(text3);
disp(text4);
disp(text5);
disp(text7);
**********************************************************************************************
```

```
********************************* Predefining the# of clusters based on selected data set **************
prompt = 'Select a number from 1 to 6  which represent your dataset :';
x = input(prompt)
if x==1;
data = xlsread('iris.xlsx');
K=3;
else if  x==2;
data = xlsread('wine.xlsx');
K=3;
else if  x==3;
data = xlsread('breastcancer.xlsx');
K=2;
else if x==4;
data = xlsread('CMC.xlsx');
K=3;
else if x==5;
data = xlsread('glass_data.xlsx');
K=6        else if x==6;
data = xlsread('CMC1.xlsx');
K=3;
else
end
end
end
end
end
end
```

```
%************************  Fetching dataset     ************************%
k=K;
data = data(1:end,:);                  %Fetch the dataset which need to be clustered
dataeva=data;                          %save the original data in dataeva
n=length (data)  ;                     %identify the data length (Number  of array Rows)
N=n;
am  = mean(data,2);                    %Compute the mean of each Row
f=numel(data)  ;                       %Identify the size of the array
width = f/n   ;                        %Identify  the # of attributes(Number  of array columns)
maxVal = max(data(:,:));               %Identify the maximum attribute value for each Dimension

S=max (maxVal);                        %Identify the maximum attribute

% **************    Normalizing the dataset in a value between [0,1]     ***************** %
for i =1:n
for h =1:width;
RANK(i,h) = S;
end
end
for i =1:n
for h =1:width;
norm_data(i,h) = (data(i,h)/RANK(i,h) );
end
end
for i =1:n
for h =1:width;
my_orig_data(i,h) = norm_data(i,h)* RANK(i,h) ;
end
end
data=norm_data;                        %saving the normalized data value  in data
```

```
************************ Experiment main loop **********************

independentRun=1;                                          %Set the number of independent runs
for d=1: independentRun;                                   %main loop ,number of runs equal  independentRun
Cset=data;
ChromosomeRepresentation=zeros(1,K);                       % Zeroing the random solution array


******** *********** Random solution selection from the dataset ******************
Perm = randperm(size(Cset,1));Perm = Perm(1:K);
% randperm function that randomize the solution ChromosomeRepresentation(1,:)=Perm;
ChromosomeRepresentation= unique(ChromosomeRepresentation,'rows');      % delete the repeated rows in
solution

************************Cluster centroid representation**********************
for i=1:K;
C(i,:)=data(ChromosomeRepresentation(1,i),:);              % loop to build the initial solution C
end
Pop=K*width;                                               % calculate the # of possible movable elements
POPULATION=Pop;
bm = mean(C,2) ;                                           % Calculate the mean of cluster centroids

********************Distance measure function (disanceindex)**********************

for i=1:K;
[E]= disanceindex(i,N,width,data,C);                       % Distance measure function(1)
distance(:,i)=E(1:end,i);
end
[M,I] = min(distance,[],2);                                % Identify the closest cluster to each object
IDX=I;                                                     % Assignment of the objects to clusters[IDX]
[clustercounter,clustersum]=newgoodness(k,width,IDX,C,am,bm,data,n);    %Function to calculate the # of objects
in each cluster
CK(1,:)=clustersum(:,1:end);
```

```
%************************ MSE function (Ggoodness) ***************************%

for i=1:K;                                                  %loop to calculate the MSE value for (k) clusters
sum4=0;
clusterindices = find(clustercounter(1:end,i));             %Identify the index of objects  belong to cluster (i)
[MSE]=Ggoodness(sum4,i,width,IDX,C,data,n,clustercounter,clustersum,clusterindices);   % MSE function(2)

ClustersMSE(1,i)=MSE;                                       %Save the MSE value for Cluster (i)
ClustersFitness(1,i)=ClustersMSE(1,i);                      %Save the MSE value for Cluster (i)

%********* Attribute cost function (costatr) ************%

for m =1:width;                                             %Inner loop to calculate the attribute cost
[MSEAtr]=costatr(i,m,C,data,clusterindices,clustersum);   % Attribute cost function  function(3)
Attribute_cost(i,m)=MSEAtr;                                 % Save the cost of attribute  (m)
end
end

MSE=sum(ClustersMSE);                                       % MSE sum for all clusters

%********************    Loop to build attribute cost matrix     *********************%
for i=1:K;                                                  %Loop to arrange the attribute value in one column
if i==1;
for m =1:width;
attribute(m,1)=C(i,m);
attribute(m,2)=i;
end
elseif i~=1;
for m =1:width;
attribute(((i-1)*width)+m,1)=C(i,m);
attribute(((i-1)*width)+m,2)=i;
end
else
end
end
for i=1:K;                                                  % Second loop to assign the cost for each attribute
```

```
if i==1;
for m =1:width
attributefitness(m,1)=Attribute_cost(i,m);
end
elseif i~=1;
for m =1:width
attributefitness(((i-1)*width)+m,1)=Attribute_cost(i,m);
end
else
end
end

for j=1:Pop;                                    %Loop to add attributes and cost to the main matrix of experiment
ChromosomeArray(j,1)=attribute(j,1);
ChromosomeArray(j,2)=attributefitness(j,1);
ChromosomeArray(j,3)=attribute(j,2);
end
NDATA = mat2gray(ChromosomeArray(:,2));   %Normalizing the cost to a value between [0-1]

for j=1:Pop;
ChromosomeArray(j,4)=1-NDATA(j,1);        % Cost transformation to fitness  (Evaluation stage for initial solution)
End
Fmax=max(ChromosomeArray (:,end));
Faverage=sum(ChromosomeArray (:,end))/Pop;
ChromosomeArraycurrent=ChromosomeArray(:,1:3);
FF1=size(ChromosomeArraycurrent);
width11=FF1(1,2);
MSE0=Fitness /K ;
fmax=max(ChromosomeArraycurrent(:,2)) ;
Fitness0=1/MSE0;
k=K;
Bestsolution = 100000;
Best=0;
```

```
%********************** SimE algorithm main loop  ***************************%

Algorihmrun=500000;                    % set the number of iterations
for i =1:width;                        % determine lower and upper value of attribute
upper_lower(1,i) =max(data(:,i));
upper_lower(2,i)=min(data(:,i));
end
R=0.95;                                % set the probability of rejecting bad moves
W=2;                                   % set number of trials
badmoveacceptance=R;
Badaccept=5;                           % set maximum accepted  drop value of bad move
B=-0.1;                                %set the bias value of selection operation β

%******************************** Iterations start ******************************%
for Run = 1 : Algorihmrun ;

%***************************SELECTION STAGE ********************************%

for i = 1:POPULATION;                          % Fetch fitness value for all attributes
mutationprobability(i,1)=ChromosomeArray(i,4);
end
for i=1:POPULATION ;       %Total number of attributes
Random=rand;
if Random>=mutationprobability(i,1)+B          %selection Based on fitness value and uniformly distributed
random numbers in the range of [0, 1],
selectionset(i,1)=1;
else
selectionset(i,1)=0;
end
end
countindicesPS=find(selectionset);             % Indices of selected attributes
countindicesPR=find(selectionset==0);          % Indices of remaining  attributes
COUNT=size(countindicesPS);
countPS=COUNT(1,1) ;                           % Selection set counter
COUNT1=size(countindicesPR);
countPR=COUNT1(1,1);                           % Remaining  set counter
```

```
for PS=1: countPS;
PSset(PS,:)=ChromosomeArraycurrent(countindicesPS(PS,1),:) ;  % fetching the selected attribute for main matrix
end
for PR=1: countPR;
PRset(PR,:)=ChromosomeArraycurrent(countindicesPR(PR,1),:);   % fetching the remaining attributes from
the main matrix
end

%**************************     ALLOCATION STAGE      ****************************%

for mutation =1:countPS ;                    %Main loop for allocation from 1st selected attribute till the last one


Sel=PSset(mutation,:);
T= find(clustercounter(1:end,Sel(:,end)));
t=clustersum(1,Sel(:,end));
dimension=countindicesPS(mutation,1)- ((Sel(:,end)-1)*width);
judger=ChromosomeArraycurrent(countindicesPS(mutation,1),2);
for Trials=1:W;
[trialsset,Probability]=SIMEMutationoperator3(Sel,K,width,N,data,T,t,Trials,upper_lower,dimension); %Mutation
function(4)
output(Trials,1:2)=trialsset(:,1:end);
output(Trials,3)=Probability;
end
output=sortrows(output,2);
for i =1:W
output(i,4)=100*((output(i,2)-judger)/(judger+output(i,2)));       % Equation to compute  drop value of bad move
end
tester= find(output(:,4)<=Badaccept & output(:,4)>0 );
tester1=size(tester);
tester1=tester1(1,1);
if ChromosomeArraycurrent(countindicesPS(mutation,1),2)> output(1,2);

ChromosomeArraycurrent(countindicesPS(mutation,1),1:2)=output(1,1:2);              % 1st case to accept good move
else
end
```

```
prob=rand;
if  tester1>=1 & prob>=badmoveacceptance;
[badsolution]=badmove2(output,width,Badaccept)  ;
ChromosomeArraycurrent(countindicesPS(mutation,1),1:2)=badsolution;      % 2nd  case to accept bad move
else
ChromosomeArraycurrent(countindicesPS(mutation,1),1:2)=ChromosomeArraycurrent(countindicesPS(mutation,1)
,1:2);                                                                                  % 3rd  case no change

end
end


%**** Centroids update, object reassignment and attribute cost calculation *********%

for i=1:K;
Cident= find(ChromosomeArraycurrent(:,end)==i);

for g = 1:width;
C1(i,g)=ChromosomeArraycurrent(Cident(g,1));     %update new centroids values after mutation process
end
end
for i=1:K;
[E]= disanceindex(i,N,width,data,C1);                    %Function to  recalculate the new distance
distance(:,i)=E(1:end,i);
end
[M,I] = min(distance,[],2);                                 % Identify the closest cluster to each object

IDX=I  ;                                                          % Assignment of the objects to clusters [IDX]

[clustercounter,clustersum]=newgoodness(k,width,IDX,C1,am,bm,data,n);  %Function to calculate the # of
objects in each cluster
for i=1:K;
clusterindices = find(clustercounter(1:end,i));             %Identify the index of objects  belong to cluster (i)
for m =1:width;
[MSEAtr]=costatr(i,m,C1,data,clusterindices,clustersum);         %Attribute cost function
Attribute_cost(i,m)=MSEAtr;       % Save the cost of attribute  (m)
```

```
end
end

%************************Build new solution matrix************************** %

for i=1:K;
if i==1;
for m =1:width
attribute(m,1)=C1(i,m);
attribute(m,2)=i;

end
elseif i~=1;
for m =1:width
attribute(((i-1)*width)+m,1)=C1(i,m);
attribute(((i-1)*width)+m,2)=i;
end
else
end
end
for i=1:K;
if i==1;
for m =1:width
attributefitness(m,1)=Attribute_cost(i,m);
end
elseif i~=1;
for m =1:width
attributefitness(((i-1)*width)+m,1)=Attribute_cost(i,m);
end
else
end
end

for j=1:Pop;
ChromosomeArray(j,1)=attribute(j,1);
```

```
ChromosomeArray(j,2)=attributefitness(j,1);
ChromosomeArray(j,3)=attribute(j,2);

end
NDATA = mat2gray(ChromosomeArray(:,2));          %Normalizing the cost to a value between [0-1]

for j=1:Pop;

ChromosomeArray(j,4)=1-NDATA(j,1);                % Cost transformation to fitness  (Evaluation operation for initial
solution)

end

Fmax=max(ChromosomeArray (:,end));                % calculate the maximum fitness
Faverage=sum(ChromosomeArray (:,end))/Pop;        % calculate  the average  fitness

ChromosomeArraycurrent=ChromosomeArray(:,1:3);    % update the main solution matrix
FF1=size(ChromosomeArraycurrent);


**********De-normalize current solution centroids value and calculate MSE value **************

for i =1:K;
for h=1:width;

Ceva(i,h)=C1(i,h)*RANK(i,h) ;                     % De-normalize cluster centroids values (original value)
end
end
for i=1:K;
[E]= disanceindex(i,N,width,dataeva,Ceva);        % Distance calculation with original dataset values(dataeva)
distance(:,i)=E(1:end,i);
end


[M,I] = min(distance,[],2);                       % identify the closest cluster to each object
IDXeva=I;                                         % Assignment of the objects to clusters [IDXeva]
```

```
[clustercountereva,clustersumeva]=newgoodnesseva(K,IDXeva,Ceva,n); );  %Function to calculate the # of
objects in each cluster
for i=1:K;
sum4=0;
clusterindiceseva = find(clustercountereva(1:end,i)); %Identify the index of objects  belong to cluster (i)

[MSE]=Ggoodness(sum4,i,width,IDXeva,Ceva,dataeva,n,clustercountereva,clustersumeva,clusterindiceseva);
ClustersMSE(1,i)=MSE;                    %Save the MSE value for Cluster (i)
ClustersFitness(1,i)=ClustersMSE(1,i);
end
MSEeva=sum(ClustersMSE);                          % MSE sum for all clusters
Simesolution(Run,1)=K/sum (ChromosomeArray(:,2));

Simesolution(Run,2)=sum (ChromosomeArray(:,2))/K;
Simesolution(Run,3)=MSEeva;


%***********This section to save the Best solution set obtained till iteration *****************%


if   Simesolution(Run,3)<Bestsolution;                 % if the current solution better than previous ones
Bestsolution=Simesolution(Run,3);                % save the MSE value of best one
BestC=Ceva;                                      % save the K centroids value
Best=Run;                                        % save the number of iteration
BestIDX=IDXeva;                                  % save object assignment
ChromosomeArrayBest= ChromosomeArraycurrent;   % save the solution matrix
BestChromosomeArray=ChromosomeArray;
BestFmax=Fmax;                                   % save best fitness value
BestFaverage=Faverage;                           % save average fitness
Bestclustercounter= clustercounter;              % save object distribution
Bestclustersum=clustersum;                       % save number of objects in each cluster
Bestclusterindices=clusterindices;               % save indices of object assigned to clusters
for i =1:K;
for h=1:width;
```

```
BestCdenorm(i,h)=CI(i,h)*RANK(i,h) ;                    % save the de-normalized values of centroids
end
end

else

end
Bestsolution;                                  % print the Best value of MSE at the end of each iteration
Best   ;                                       % print the iteration number which have Best MSE value
if Best==Run;                                  % counter to calculate  how many iterations improved the solution quality
representI(Run,1)=Best;
representI(Run,2)=Bestsolution;
else
representI(Run,1)=0;
representI(Run,2)=0;
end

%*****Starting from the best solution if there is not improvement over 100 iterations *******%

SSEeva;
if Run-Best>= 100
ChromosomeArraycurrent=ChromosomeArrayBest;
ChromosomeArray= BestChromosomeArray;
Fmax=BestFmax;
BestFaverage=BestFaverage;
clustercounter=Bestclustercounter;
clustersum=Bestclustersum;
clusterindices=Bestclusterindices;
else
end

end                    %End of algorithm loop
%********************* Independent run  MSE value  ********************************%


inDependent_1(d,1)=Bestsolution;                    % Save the best MSE value for each independent Run
```

```
fprintf('MSE= %d .\n',inDependent_1(d,1));              % Print the best MSE value for each independent Run


for i=1: Algorihmrun ;
y(i,1)=1/(Simesolution(i,3));
end


algorithaverage=(sum (inDependent_1(:,:))) /  independentRun ;          % Calculate the average MSE value of all
independent run

disp('MSE value ');
disp(inDependent_1) ;
```

Functions code:

```
%************ disanceindex Function for Euclidean distance measure***************%

function [E]= disanceindex(i,N,width,data,C);    % Function to calculate the Euclidean  distance between cluster and
object


for m=1:N;
ecl=0;
for h =1:width ;                                 %From the first attribute to the last attribute (dimension)
ecl = ecl+ (data(m,h)- C(i,h))^2;
end
E(m,i)=(ecl);
end

%********************** (Ggoodness) function for MSE calculation**********************%
```

```
function [MSE]=Ggoodness(sum4,i,width,IDX,C,data,n,clustercounter,clustersum,clusterindices);

for j=1:(clustersum(1,i));                % within the cluster calculation for all the data points assigned to this cluster
ecl=0;
for h =1:width ;                          %from the first dimension up to the last dimension
z=  clusterindices(j,1);
ecl = ecl+ ((data(z,h)- C(i,h))^2);
end                                       % end of the inner loop after completing this one data point MSE will be computed
sum4=sum4+ sqrt (ecl) ;
end                                       % after this point one cluster MSE value is completed
MSE= sum4 ;

%********************(costatr) function for calculating the attribute cost*****************%
function [MSEAtr]=costatr(i,m,C,data,clusterindices,clustersum);
Total=0;
for s=1:(clustersum(1,i));                % total number of objects assigned to this cluster
Total = Total+ (data(clusterindices(s,1),m)- C(i,m))^2; % calculate the distance between the object attribute and
centroid attribute
end
MSEAtr=Total;
%**** (SIMEMutationoperator3) function for altering the attribute values  Mutation operation*****%
function [trialsset,Probability]=SIMEMutationoperator3(Sel,K,width,N,data,T,t,Trials,upper_lower,dimension);
a=1;
b=0;
valid=0;
count1=0;
random2=rand;
if random2<=0.5                           % based on probability to decide addition or subtraction operation
Random1 = (a + (b-a).*rand)/50;   % calculate the positive value
```

```
else
RandomI = -(a + (b-a).*rand)/50;   % calculate the negative value
end
offspring= Sel(1,1)+RandomI;                    % calculate the new value of attribute
upper=upper_lower(1,dimension);
lower=upper_lower(2,dimension);
clustersum=t;
clusterindices=T;
[SSEAtr]=costatr1(dimension,offspring,data,T,t);  % calculate the new attribute cost
trialsset(1,1)=offspring;                                    % save the attribute value
trialsset(1,2)=SSEAtr;                                       % save the attribute cost
Probability=random2;                                         % save the probability value
```

REFERENCES

[1] http://www.dictionary.com/browse/clustering

[2] Bora, Mr, et al. "Effect of different distance measures on the performance of K-means algorithm: an experimental study in Matlab." arXiv preprint arXiv:1405.7471 (2014).

[3] Aggarwal, Charu C., and Chandan K. Reddy, eds. Data clustering: algorithms and applications. CRC press, 2013.

[4] Lu, Yi, et al. "FGKA: A fast genetic k-means clustering algorithm." Proceedings of the 2004 ACM symposium on Applied computing. ACM, 2004.

[5] Hruschka, Eduardo Raul, Ricardo JGB Campello, and Alex A. Freitas. "A survey of evolutionary algorithms for clustering."IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39.2 (2009): 133-155.

[6] https://en.wikipedia.org/wiki/Mean_square_quantization_error

[7] https://en.wikipedia.org/wiki/Mean_squared_error

[8] Yao, Andrew Chi-Chih. "Some complexity questions related to distributive computing (preliminary report)." Proceedings of the eleventh annual ACM symposium on Theory of computing. ACM, 1979.

[9] Papadimitriou, Christos H., and John Tsitsiklis. "On the complexity of designing distributed protocols." Information and Control 53.3 (1982): 211-218.

[10] https://en.wikipedia.org/wiki/Similarity_measure

[11] Gomaa, Wael H., and Aly A. Fahmy. "A survey of text similarity approaches." International Journal of Computer Applications 68.13 (2013).

[12] Linoff, Gordon S., and Michael JA Berry. *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 2011.

[13] Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis. "Clustering validity checking methods: part II." ACM Sigmod Record 31.3 (2002): 19-27.

[14] Rendón, Eréndira, et al. "Internal versus external cluster validation indexes." International Journal of computers and communications 5.1 (2011): 27-34.

[15] Maulik, Ujjwal, and Sanghamitra Bandyopadhyay. "Genetic algorithm-based clustering technique." Pattern recognition33.9 (2000): 1455-1465.

[16] Krishna, K., and M. Narasimha Murty. "Genetic K-means algorithm." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29.3 (1999): 433-439.

[17] Lu, Yi, et al. "Incremental genetic K-means algorithm and its application in gene expression data analysis." BMC bioinformatics 5.1 (2004): 172.

[18] Handl, Julia, and Joshua Knowles. "An evolutionary approach to multiobjective clustering." IEEE transactions on Evolutionary Computation 11.1 (2007): 56-76.

[19] Bandyopadhyay, Sanghamitra, Anirban Mukhopadhyay, and Ujjwal Maulik. "An improved algorithm for clustering gene expression data." Bioinformatics 23.21 (2007): 2859-2865.

[20] Lam, Yau-King, Peter Wai-Ming Tsang, and Chi-Sing Leung. "PSO-based K-Means clustering with enhanced cluster matching for gene expression data." Neural Computing and Applications 22.7-8 (2013): 1349-1355.

[21] Sun, Shiquan, and Qinke Peng. "A hybrid PSO-GSA strategy for high-dimensional optimization and microarray data clustering." Information and Automation (ICIA), IEEE International Conference on. IEEE, 2014.

[22] Hatamlou, Abdolreza, Salwani Abdullah, and Hossein Nezamabadi-Pour. "A combined approach for clustering based on K-means and gravitational search algorithms." Swarm and Evolutionary Computation 6 (2012): 47-52.

[23] Zhang, Changsheng, Dantong Ouyang, and Jiaxu Ning. "An artificial bee colony approach for clustering." Expert Systems with Applications 37.7 (2010): 4761-4767.

[24] Liu, Yongguo, et al. "A tabu search approach for the minimum sum-of-squares clustering problem." Information Sciences178.12 (2008): 2680-2704.

[25] Güngör, Zülal, and Alper Ünler. "K-harmonic means data clustering with tabu-search method." Applied Mathematical Modelling 32.6 (2008): 1115-1125.

[26] Chang, Dong-Xia, Xian-Da Zhang, and Chang-Wen Zheng. "A genetic algorithm with gene rearrangement for K-means clustering." Pattern Recognition 42.7 (2009): 1210-1222.

[27] Bandyopadhyay, Sanghamitra, Ujjwal Maulik, and Malay Kumar Pakhira. "Clustering using simulated annealing with probabilistic redistribution." International Journal of Pattern Recognition and Artificial Intelligence 15.02 (2001): 269-285.

[28] Chen, Ching-Yi, and Fun Ye. "Particle swarm optimization algorithm and its application to clustering analysis." Electrical Power Distribution Networks (EPDC), Proceedings of 17th Conference on. IEEE, 2012.

[29] Van der Merwe, D. W., and Andries Petrus Engelbrecht. "Data clustering using particle swarm optimization." Evolutionary Computation, CEC'03. The 2003 Congress on. Vol. 1. IEEE.

[30] Xiao, Xiang, et al. "Gene clustering using self-organizing maps and particle swarm optimization." Parallel and Di*stributed Processing Symposium,* Proceedings. International. IEEE, 2003.

[31] Cui, Xiaohui, Thomas E. Potok, and Paul Palathingal. "Document clustering using particle swarm optimization."Swarm Intelligence Symposium, Proceedings IEEE, 2005.

[32] Hatamlou, Abdolreza, Salwani Abdullah, and Masumeh Hatamlou. "Data clustering using big bang–big crunch algorithm." Innovative Computing Technology. Springer, Berlin, Heidelberg, 2011. 383-388.

[33] Hatamlou, Abdolreza. "Black hole: A new heuristic optimization approach for data clustering." Information sciences 222 (2013): 175-184.

[34] Prakash, Jay, and Pramod Kumar Singh. "Evolutionary and swarm intelligence methods for partitional hard clustering."Information Technology (ICIT), International Conference on. IEEE, 2014.

[35] Das, Swagatam, Ajith Abraham, and Amit Konar. "Automatic clustering using an improved differential evolution algorithm."IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans 38.1 (2008): 218-237.

[36] Selim SZ, Alsultan K. A simulated annealing algorithm for the clustering problem. Pattern recognition. 1991 Jan 1;24(10):1003-8.

[37] Sait, Sadiq M., and Habib Youssef. "Iterative Computer Algorithms: and their applications in engineering." (1999).

[38] Sait, Sadiq M., Mustafa I. Ali, and Ali Mustafa Zaidi. "Multiobjective VLSI cell placement using distributed simulated evolution algorithm." ISCAS 2005, IEEE International Symposium on Circuits and Systems.

[39] Sait, Sadiq M., and Junaid A. Khan. "Simulated evolution for timing and low power VLSI standard cell placement."Engineering Applications of Artificial Intelligence 16.5 (2003): 407-423.

[40] Bangerth, W., et al. "On optimization algorithms for the reservoir oil well placement problem." Computational Geosciences 10.3 (2006): 303-319.

[41] Černý, Vladimír. "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm." Journal of optimization theory and applications 45.1 (1985): 41-51.

[42] C.L. Blake, C.J. Merz, UCI repository of machine learning databases. Available from: http://www.ics.uci.edu/-mlearn/MLRepository.html

[43] https://measuringu.com/statistically-significant

Vitae

Name: Ahmed Mohamed Elbadawi Abdelsatir

Nationality: Sudanese

Date of Birth: March 19, 1981

Email: abadawi@saudia.com

Address: Al-Khobar, Kingdom of Saudi Arabia